

# Knowledge Engineering for Embedded Configuration

**Oddsson, Gudmundur Valur**

*Publication date:*
2008

*Document Version*
Publisher's PDF, also known as Version of record

# Knowledge Engineering for Embedded Configuration

A PhD thesis by
Guðmundur Valur Oddsson

PhD Thesis at the Technical University of Denmark
Department of Management Engineering

**Supervisor:** Professor Lars Hvam
Department of Management Engineering
Technical University of Denmark

**Opponents:** Professor Niels Henrik Mortensen
Department of Management Engineering
Technical University of Denmark

Professor Johan Malmqvist
Department of Product and Production Development
Chalmers University of Technology

Staffan Sunnersjö
Professor Machine Design
School of Engineering, Jönköping University

The presented thesis is a part of the acquisition of a Ph.D. degree

Title: Knowledge Engineering for Embedded Configuration



Published by:
Department of Management Engineering
Technical University of Denmark
2800 Kongens Lyngby
Phone: (+45) 4525 2525

ISBN: 978-87-90855-19-2

# ABSTRACT

This thesis presents a way to simplify setup of complex product systems with the help of embedded configuration. To achieve this, one has to focus on what subsystems need to communicate between themselves. The required internal knowledge is then structured at three abstraction levels. Simplifications of the internal workings are both due to hardware- and application-induced configuration taking place both within the overall system and in each subsystem. By relating parameters in such a way, the number of user inputs or decision variables should decrease drastically, thus increasing the overall usability of the installation. In our case, we have rationalized that this should be done with embedded configuration, and the expected result is enhanced usability.

The suggested method is deeply rooted in system theory. It draws on the emergent properties expected from the system, and tries to embed into the system the knowledge needed to achieve them. In order to understand the system, one draws simplified functional streams and identifies archetypes from the product assortment, and then one maps the two together into a system breakdown model. The system model indicates how many encapsulation models (EMs) should be made and the first decomposition in their tree centrepiece. The encapsulation model describes the archetype on three abstraction levels: application, function, and the physical artefact. All levels are connected through relational matrixes both for internal and mapping relations, and predefined relation types are suggested. The models are stringent and thought out so they can be implemented in software. They should allow both import and export of product knowledge from the knowledge-based system.

The purpose of this work is to simplify the installation process of product systems that have been treated with extreme postponement, meaning that variance is given with variables while installing. Variables here can be both software- and hardware-like in nature. These variables are defined as decision variables, and it is the reduction of these variables that is the overall goal.

The next step can be said to be two-fold: first, to construct a system based on this philosophy and to show that it actually leads to the expected results. And second, to further develop the modelling tools and methods for supporting the making of embedded configuration systems, or in essence, a distributed artificial intelligence system.

# RESUME

Denne afhandling omhandler om simplificering af opsætning af komplekse produkt systemer med indlejret konfigurering. Fo at opnå dette, bør fokus rettes mod undersystemer og hvilke funktionelle behov disse undersystemer har for at kommunikere indbyrdes. Den nødvendige information er herefter struktureret i tre abstraktion niveauer, applikation, funktion og fysisk struktur. Simplificering forgår for både hardware og applikation induceret konfigurering, både på overordnet niveau og i hvert undersystem. Ved at relatere parametre på denne måde, falder antallet af bruger inputs eller beslutningsvariable drastisk og øger dermed den overordnede brugervenlighed af installationen. I vores case, vi har givet begrundelse for at dette skulle udarbejdes med indlejret konfigurering og hvor det forventede resultat er at forbedre brugervenligheden.

Den forslåede metode er meget begrundet i system teori. Den bygger på helheds egenskaber af et system og forsøger at indlejre viden nødvendige for at opnå disse. Til at forstå et system, tegnes en forenklet funktionel strøm og der identificeres hovedtyper fra produkt assortimentet. De to er koblet sammen til et system breakdown model. System modellen indikerer hvor mange indlejringsmodeller skal udarbejdes og forslår de første nedbrydninger af modellens middelstykke. Indlejringsmodellen beskriver hovedtyper på de tre abstraktions niveauer. Alle niveauer er forbundet med relations matricer både for interne og mellem niveauer med forslåede relationstyper. Modellen er stringent og konstrueret til at blive implementeret i et IT system. Modellen skulle tillade både import og eksport af produkt viden fra et vidensbaseret system.

Formålet med dette arbejde er at forenkle installations processen af et produkt system der er behandlet med ekstrem postponement hvor varians er givet ved installation. Parametrene kan være både software og hardware orienterede. Disse variable er navngivet beslutningsvariabler og det overordnede målsætning er at reducere antallet af variable.

Det næste step er todelt: for det første at konstruere et system baseret på denne filosofi for at vise at det aktuelt giver de forventede resultater. Og for det andet, at videreudvikle modellens værktøjer og metoder til at supportere fremstilling af indlejrede konfigurations systemer, det vil sige et distribueret kunstigt intelligens system.

ACKNOWLEDGEMENT

This Ph.D. thesis is a result of a three-year labour carried out at the Department of Management Engineering, Technical University of Denmark. The project was initiated in June 2005 and completed in July 2008.

I like to say that the process was not as unpleasant as many warned it would be. The project was handled like a full-time job and the experience of dealing with inspirational and stimulating challenges associated with the process were mostly enjoyable. There were times of course when there was press, stress and thoughts like "why did I get myself into this", but mainly these were good years. However, I would not have made it if it had not been for several people that gave a helping hand in one way or another.

I therefore offer my thanks to the following persons and organizations:

- My family: my wife Anna Björg and our daughters Helga Valborg and Vala Katrín, for keeping me sane!
- My PhD colleagues, Klaes Ladeby, Anders Haug, Tim Christensen and Andreas Traberg for their pleasant company on journeys to conferences and PhD-courses, and for interesting discussions.
- The case company, Grundfos, for giving me access to their organization and supplying a very interesting problem.
- All at Institito de Empresa (IE), Madrid, Spain for allowing me to join their PhD school for a semester and especially Professor Fabrizio Salvador for accepting my plea and discussing my work, and Professor Angel Diaz for allowing me to attend the PhD school.
- To the organizations that gave me financial support to make an external stay a possibility: "Otto Mønsted Fond", "Reinholdt W. Jorck og Hustrus Fond" and "Minningarsjodur Helgu og Sigurlida".
- Professor Mogens Myrup Andresen (DTU) for taking the time to discuss the field of engineering design.
- Professor Staffan Sunnersjö for introducing me to the field of cognition research and automation in engineering design.
- Correspondent Christina Scheel Christiansen (DTU) for her enormous work aiding me in the administration of my project and for many interesting discussions on life, the universe and everything.
- My supervisor, Professor Lars Hvam, for ensuring me good conditions while I worked on my PhD, and especially for providing possibilities to test my propositions with students and supporting my choice of the directions for the research.

Finally, I also offer my thanks to all those I may have forgotten who also have played an important role in the research process.

Kgs. Lyngby, Denmark, in August 2008,

Gudmundur Valur Oddsson

GLOSSARY

| | |
|---|---|
| **PVM** | Product Variant Master |
| **PFMP** | Product Family Master Plan |
| **EM** | Encapsulation Model |
| **CM** | Communication Model |
| **SBS** | System Breakdown Structure |
| **OM** | Operation Management |
| **AR** | Action Research |
| **ED** | Engineering Design |
| **DRM** | Design Research Method |
| **KE** | Knowledge Engineering |
| **KBS** | Knowledge-Based System |
| **MFD** | Modular Function Deployment |
| **DSM** | Design Structure Matrix |
| **DMM** | Domain Mapping Matrix |
| **ESM** | Engineering System Matrix |
| **QFD** | Quality Function Deployment |
| **UML** | Unified Modelling Language |
| **MML** | MOKA Modelling Language |
| **SysML** | Systems Modelling Language |
| **FB** | Functional Basis |
| **IDEF** | Integrated DEFinition methods |
| **MFM** | Multi-Flow Modelling |
| **GFM** | Goal Function Modelling |
| **GTST** | Goal Tree – Success Tree |

TABLE OF CONTENTS

# *Chapter 1*

# INTRODUCTION

"*Systems everywhere*", as so eloquently stated by Bertalanffy (Bertalanffy 1969), is an excellent way to describe our modern society. It surrounds us with systems to help us in our daily lives and to make them better. Many of those systems are "hidden" from us, and we only notice them when they fail. Think about the facilities that one uses daily, like electricity, water, sewers and heating. Those are complex product systems that are made by combining several subsystems (or independent products) to form a whole. Many of the subsystems are equipped with computers. By setting parameters, these subsystems can be allowed a wide range of different setups, just as if they were different products. To make the subsystem work in a context, these parameters need to be set.

In order to make complex product system setups easier, an idea is brewing. If one were able to encapsulate as much product knowledge in each subsystem as possible, have internal configuration engines keep track of internal consistency, and focus communication between subsystems so that only core information or knowledge is transferred, setups would be simplified. The idea is to make the systems kind of aware, and provide them knowledge to "self-configure" once the hardware connections are made. This idea has a lot in common with the field of artificial intelligence, especially the branch of distributed artificial intelligence, and inspiration has been drawn from this field about how to go about solving it.

This thesis tries to connect three aspects of making distributed knowledge systems, namely the *encapsulation of product knowledge*, its subsequent *encoding into product models*, and finally, the *communication of knowledge between subsystems*. To achieve this, one has to look at the building blocks needed for such construction, namely: knowledge structuring, knowledge representation, communication of knowledge, and finally, modelling in such a way that it will support the concept. The solution suggested for making distributed knowledge systems work is hereafter named *embedded configuration*. Let us now start by looking at the problem and then move to the research activities that are to be addressed in this thesis.

## 1.1 THE PROBLEM IN BRIEF

Inauguration of this work is based on a problem recognized at the case company. The company is one of the world-leading manufacturers of pumps and pump systems. They have been really successful in modularizing a part of their product assortment and have utilized black boxes with multi functions that are only made to a specific variant when installing, in order to reach even greater economies of scale in production. On the other hand, this has led to embedded software in each of the black boxes and a lot of parameters that decide what parts of the functionality are used each time. Therefore,

the task of giving variance to the system has been moved out of the production hall and downstream to installation at the customer site.

An example of an installation complexity could be the water supply in a given building of reasonable size; it would require about four to six pumps, each with 500 parameters controlled by a controller (in the sense of a controlling unit, not a person) that has 3000 parameters to set. One can easily see the tediousness of installing such a system.

Appling the aforementioned idea to the example mentioned above should aid in clarifying the suggested method. When connecting a pump to a controller, some parameters have to be set to "tell" each device what it is to be connected to. The pump has to know that it is being controlled externally, and it has to "relinquish" its own control. The controller has to know the attributes of the pump, so that it can control the overall system accordingly. These things are all hardware-related in the sense that they help define the solution space, i.e. what applications are possible for the whole system. If the subsystems could communicate meaningfully, these tasks would "disappear" from the installation. When selecting a specified application, some of the devices have to be told what is being done. A part of the application is to set redundancy. What is the pump to do if the connection to the controller is lost, and vice versa, how shall the controller handle lost connection to the pumps? The next chapter will deal with the problem description in detail, so let us move on to the research involved in this practical problem.

## 1.2 RESEARCH QUESTIONS

Based on the problem, we have identified four main areas that need investigation. These areas can be crystallized into four key questions. These are the main research questions and core intake of this thesis. The four questions are:

1. *How can product knowledge be modularized in order to allow encapsulation but without losing overall system integrity?*
2. *What should the modelling technique be like in order to support modularized product knowledge?*
3. *What should the process for building embedded configuration system be like?*
4. *Why should modularized product knowledge and embedded configuration systems be implemented?*

These questions are not equal in substance. Questions one and four are much "lighter" than questions two and three. The main bulk of this thesis deals with questions two and three; however, the answer to question one is a necessary prerequisite to number two, just as the answer to number two serves as input to number three.
The research questions are very broad, and to aid the process of dealing with them, we must specify more concrete supporting questions that identify specified issues related to the overall research questions.

### 1.2.1 SUPPORTING QUESTIONS

With an outset in the four research questions, let us list several supporting questions that guide the process of dealing with the research questions. The supporting questions are numbered to refer to the research questions, even though the group names are not the same.

1. *Conceptualizing embedded configuration*
   - *1.1. What is the role of modularization in embedded configuration?*
   - *1.2. How does one encapsulate product information?*
   - *1.3. How do modules interface each other?*
   - *1.4. How do they communicate?*
   - *1.5. Is it necessary to divide communication into layers?*
   - *1.6. Who controls in a system of peers and how?*
   - *1.7. How are new versions / upgrades of modules handled?*
   - *1.8. How does one ensure information flow (no redundancy)?*

2. *Modelling product knowledge to support embedded configuration*
   - *2.1. How does one encapsulate variance (hiding internal parameters)?*
   - *2.2. How does one tie functions and structure together?*
   - *2.3. How is the model interfaced?*
   - *2.4. How does communication function between PVMs?*
   - *2.5. What kind of complexity is needed in the model?*
   - *2.6. How is dynamic communication between different PVMs allowed?*
   - *2.7. Where is product data stored?*
   - *2.8. Where are rules of combination stored?*
   - *2.9. Where are rules implemented?*

3. *Developing embedded configuration systems*
   - *3.1. How does one decompose an embedded configuration system?*
   - *3.2. How does one make the models?*
   - *3.3. How does one relate the models to the environment?*
   - *3.4. How does one acquire data to populate the models?*
   - *3.5. What are the success factors of embedded configuration?*

The research approach is an iterative process of analysing, finding solution bits, collecting the snippets, melting them together, and them trying out. We discuss the research approach in more detail later in this chapter. To summarize, the research approach is an iteration of the following activities:

- Analysing the current situation at Case Company to clearly identify the problem, and later to test the suggestions.
- Searching the literature, in a broad sense, for possible solution bits, analogies and eventual solutions to similar problems in other areas.
- Incorporating the snippets into current DTU methodology for construction of configuration systems.
- Constructing a solution suggestion based on the aforementioned activities.

It is worth noticing that this iteration cycle has dual goals: the proposed solution to a specific problem (situation), and theoretical advancement of the DTU methodology. Or to rephrase the duality of the goals: the goals are to reduce complexity for after-sales

service, and to create a modelling technique to support this. Let us now look at the research method applied.

## 1.3 RESEARCH METHOD

Doing research is not only solving problems, it is also about being scientific. In this section, we look very briefly at the research method used and what it entails. There are three aspects to the process: the method used, field of interest (e.g. the domain in question), and the worldview of science used in the approach. It is not the intention of this section to present an in-dept discussion of research methods and worldviews. For such a discussion in relation to operation management, see my colleague's work on *Representation of Industrial Knowledge - as a Basis for Developing and Maintaining Product Configurators* by Anders Haug (Haug 2007). Here, we look at the three aspects mentioned above and state our standpoint regarding each. But first, we need to look at two things: qualitative research and theory.

This work has qualitative aspects, as it is not derived through the application of statistics. Qualitative research is better explained in the following quote:

> *By the term qualitative* research *we mean any kind of research that produces findings not arrived at by means of statistical procedures or other means of quantification.*
> *(p. 17) in (Strauss & Corbin 1990)*

As a consequence of the fact that this research is a single-case research, the empirical base used here is not large enough to use statistics. The results are generated through fieldwork and as a solution to a specific problem. Then, there is theory. What is theory? Is this work about generating theory? To discuss this, we need to know what theory is.

> *Generally, academics point to a theory as being made up of four components, (1) definitions of terms or variables, (2) a domain where the theory applies, (3) a set of relationships of variables, and (4) specific predictions (factual claims).*
> *(p.363) in (Wacker 1998)*

This work easily complies with the first three points, but at present, it is a bit lacking in predictions. Although we feel confidant that this work can be structured to form a theory, that is not the purpose here. We lean towards fact finding research (Wacker 1998) where sense is made of actual problems. Fact-finding research collects data and starts the journey towards theory making by contributing to the first three components of theory. The work here could probably be described as grounded theory (Strauss & Corbin 1990).

> *A **grounded theory** is one that is inductively derived from the study of the phenomenon it represents.*
> *(p.23) in (Strauss & Corbin 1990)*

The formalization needed to make this work into theory is as mentioned earlier is not the issue here, but we recognize the way to go. Excellent work on how to build theory

in Operation Management (OM) through cases and field research (Meredith 1998) and how to make a "good" theory in OM (Wacker 2004) could lead the way to learning how to arrive at a theory. We discuss this later in this thesis, when we formulate our first attempt at theory (see section 8.5 on page 198). Let us now return to the three aspects mentioned above: the research method, the domain view, and the worldview of science, and look at them in that order.

### 1.3.1 THE RESEARCH METHOD

This work has its root in operation management. Observing a problem in the case company, analysing it and suggesting a solution has triggered the work presented here. Analysis indicates that the problem can be traced back to the product design. So, we need to solve an operation problem while suggesting changes in the engineering design. This also means that the research method applied will draw on both the field of Operation Management (OM) and Engineering Design (ED). A summary of methods applied in OM is shown in Figure 1.

| | NATURAL ⟷ ARTIFICIAL | | |
|---|---|---|---|
| RATIONAL | DIRECT OBSERVATION OF OBJECT REALITY | PEOPLE'S PERCEPTIONS OF OBJECT REALITY | ARTIFICIAL RECONSTRUCTION OF OBJECT REALITY |
| AXIOMATIC | | | • REASON/LOGIC/ THEOREMS<br>• NORMATIVE MODELING<br>• DESCRIPTIVE MODELING |
| LOGICAL POSITIVIST/ EMPIRICIST | • FIELD STUDIES<br>• FIELD EXPERIMENTS | • STRUCTURED INTERVIEWING<br>• SURVEY RESEARCH | • PROTOTYPING<br>• PHYSICAL MODELING<br>• LABORATORY EXPERIMENTATION<br>• SIMULATION |
| INTERPRETIVE | • ACTION RESEARCH<br>• CASE STUDIES | • HISTORICAL ANALYSIS<br>• DELPHI<br>• INTENSIVE INTERVIEWING<br>• EXPERT PANELS<br>• FUTURES/ SCENARIOS | • CONCEPTUAL MODELING<br>• HERMENEUTICS |
| CRITICAL THEORY | | • INTROSPECTIVE REFLECTION | |
| EXISTENTIAL | | | |

**Figure 1 - A framework for research methods (Meredith 1989)**

An explanation of the axis and terms on each side, the Natural-Artificial axis and Rational-Existential axis, is shown in Figure 2. This work is actually placed in two places in this framework (Figure 1): action research to deal with the OM part, and conceptual model to deal with the ED part. A knowledge transfer takes place between the two.

Because of qualitative aspects and the fact that this research is not passive in nature, we observe that this is action research (AR). Action research in operation is described in (Coughlan & Coghlan 2002). The trademarks of AR can be summarized in two points:

- Dual goals: solve a practical problem and add to the body of knowledge
- Active and not passive method. Tight cooperation with case company with the researcher trying to lead / influence the process, in contrast to observing without interference

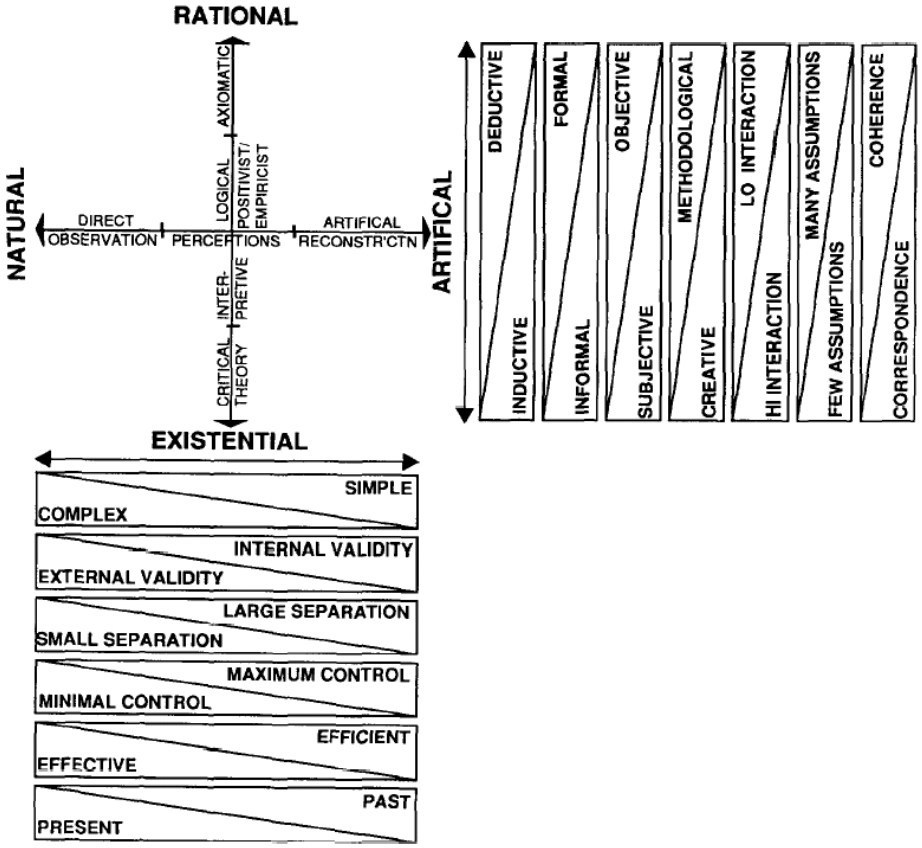



**Figure 2 - A generic research framework (Meredith 1989)**

A comparison between conventional positivistic research and AR is shown in Figure 3.

| | Positivist science | Action research |
|---|---|---|
| Aim of research | Universal knowledge<br>Theory building and testing | Knowledge in action<br>Theory building and testing in action |
| Type of knowledge acquired | Universal<br>Covering law | Particular<br>Situational<br>Praxis |
| Nature of data | Context free | Contextually embedded |
| Validation | Logic, measurement<br>Consistency of prediction and control | Experiential |
| Researcher's role | Observer | Actor<br>Agent of change |
| Researcher's relationship to setting | Detached neutral | Immersed |

**Figure 3 - Comparison of positivist science and AR (Coughlan & Coghlan 2002)**

This framework fits perfectly with the work done here. Actually, this is very much in agreement with the so-called *Nordic approach* in research (Jørgensen 1992). This

approach also preaches the duality of research done in Scandinavia: solving a practical problem while adding to the body of knowledge. This approach is shown in Figure 4.

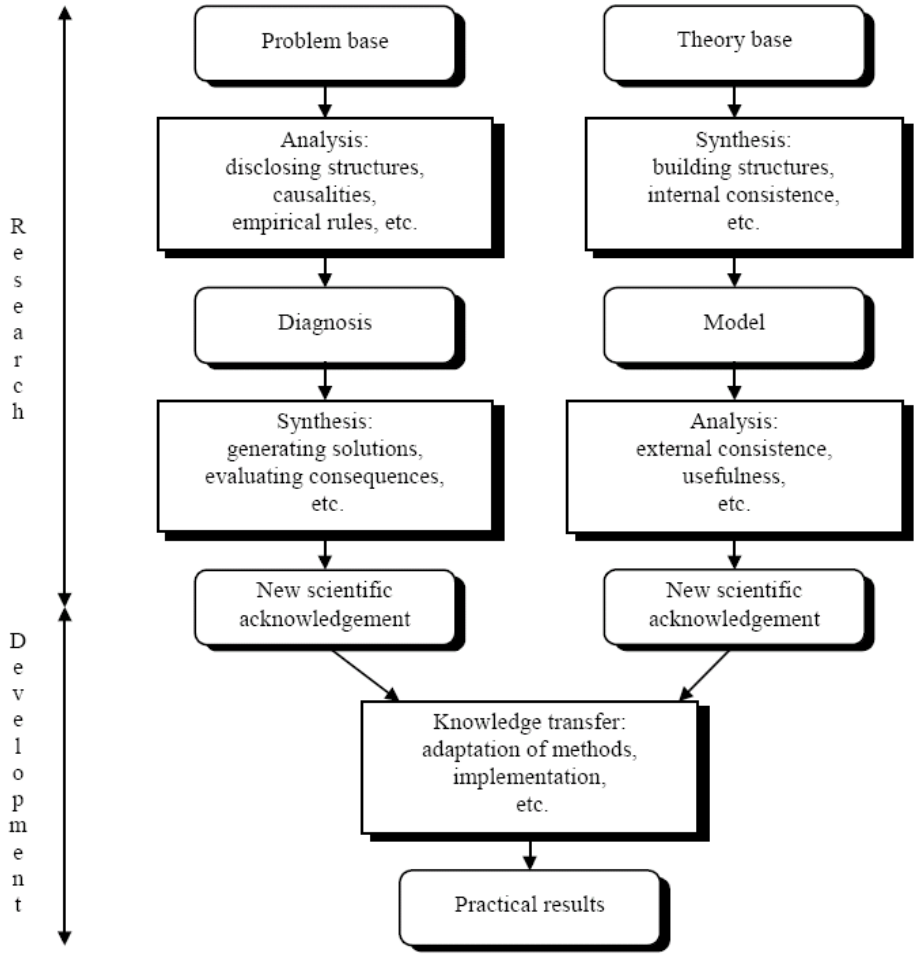



**Figure 4 - Scientific approach (Jørgensen 1992)**

Much of the work done at our institute (DTU Management Engineering) has followed the Nordic approach, and so will we. Next, let us look at the domain view.

### 1.3.2 THE DOMAIN VIEW

Duality is the word to use when talking about the domain in question. As mentioned earlier, there are two domains involved here, operation management and engineering design. They have separate goals, but one is a prerequisite in order to reach the other. The overall goal is the OM goal: to *reduce complexity* in installation and maintenance of complex product setups. To reach this goal, we need to redesign the complex product system. These redesigns require some engineering design work and to facilitate this we come to the goal of ED: *formalized modelling technique complete with ontology for relationship, decomposition and taxonomy*. These two goals coincide with the duality mentioned in previous section; the complexity goal is tackled by using action research, while the modelling goal is the conceptual model. We have discussed OM research methods. Let us now look at such methods for engineering design.

This thesis uses DRM (Design Research Method) suggested by (Blessing 2002). This method formalizes the process of looking at design and provides structure for iterations.

DRM interlocks nicely with the overall OM framework and AR via the *criteria* stage shown in Figure 5.

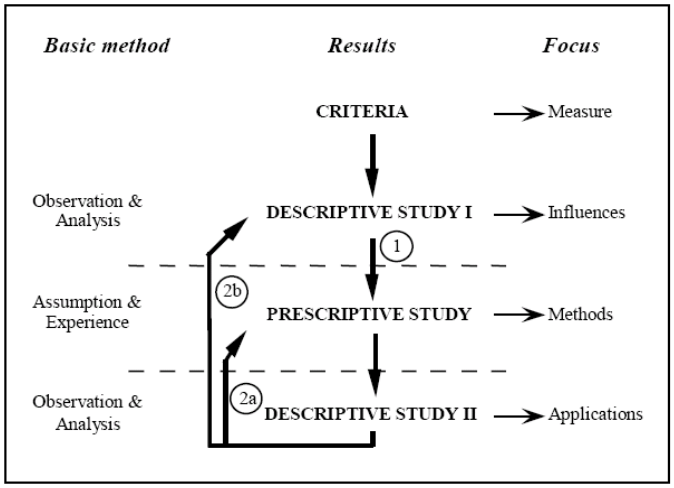



**Figure 5 - DRM (Design Research Method) framework (Blessing 2002)**

These two frameworks are actually complementary. Let us finally look at the different worldviews of sciences.

### 1.3.3 THE WORLD VIEW USED

What are science, knowledge and progress? These are some of the issues that philosophers have discussed throughout the ages. The classical Popper vs. Kuhn discussion is not up for debate here. We do not go into the different *isms* (again for detailed discussion, see (Haug 2007)). What we need to know is that the two main schools, *Critical Rationalism* (Popper 2002) and *Constructivism* (Kuhn 1996), differ on how the world is viewed. The former states that all theories must be considered untrue, which means that science must be performed through an endless series of revisions. New theories replace old ones because they explain things better. The scientific method of critical rationalism is thus a cyclic trial-and-error process. The constructivist view of the world states that there is a "filter" between the subject and reality. Reality is not independent of the subject but is socially "constructed" by people through their perceptions.

Most research done at our institute relies on critical rationalism and falsification for its worldview. On the other hand, system theory relies on constructivism to explain its world. It states that:

> *... the view is adopted that 'Knowledge' is the conceptual means to make sense of experience, rather than a 'representation' of something that is supposed to lie beyond it ... (p.237) in (Glasersfeld 1991)*

This is in concord with the work done here. It is important to realize that the view of constructivism is not general but specific to knowledge or better stated:

> *From my point of view, the trouble is that most critics seem to be unwilling to accept the explicit, programmatic statement that constructivism is a theory of knowing, not of being. (p41) in (Glasersfeld 2001)*

In this thesis, where we use the previous work done at our institute together with system theory and its application, a philosophical question seems to rise. As the fields adhere to different worldviews, what view should be selected here? Or are these worldviews not mutually exclusive? We do not attempt to answer this question but leave it for others to discuss.

### 1.3.4 RESEARCH METHOD USED

The previous sections have discussed some relevant issues of doing research. To summarize and to explicitly state our standpoints, let us review the three aspects and state our stand. This is a qualitative research in two domains, operation management and engineering design. The research method used is a merger of the methods from the two domains and the current paradigm, the Nordic approach. The overall method is AR with particular inclusion of the DRM to deal with the conceptual model suggested. Another way of putting it is to say that AR governs the problem track in the Nordic approach, while the theory track uses DRM. On the issue of worldview, we do not take an explicit stand. We acknowledge the merit of each view and recognize that our research actually uses both (if that is possible). The closest description could be that the problem track is governed by critical rationalism and the theory track by constructivism. But again, we leave that discussion to others.

## 1.4 LIMITATIONS AND FOCUS

The work presented in this thesis adheres to some severe limitations. Although the field of knowledge engineering uses a five-step method (Kendal & Creen 2007) where the steps are: *Knowledge Acquisition*, *Knowledge Validation*, *Knowledge Representation*, *Inference* and *Explanation and Justification*, this thesis focuses only on the third step *Knowledge Representation*! The thesis title could be "Knowledge Representation for Embedded Configuration", but it has been decided to use the more general term "Knowledge Engineering", in spite of the previously mentioned limitation.

### 1.4.1 ASSUMPTIONS

The second thing to mention is that in this work, many assumptions are made along the way. These assumptions are also very limiting and could for that matter be called limitations. They are best summarized as bullets:

- One assumes a combinatory product system where final setup is not known beforehand.
- A computer is used in most (if not all) sub-systems.
- This work applies only to rule-based configuration (Stumptner 1997).
- This is not work on protocol; we assume an open communication channel with an arbitrary protocol. In other words, this work is about the meaning of communication, not the technical implementation.

- The product system has emergent properties (Checkland 1984). This means the system properties are not embedded in each sub-system but only arise when the whole is assembled.
- The overall system exhibits "equifinality" (Bertalanffy 1969) or purposefulness (Ackoff 1971).
- Relations are considered dyadic in nature (binary).

These assumptions are apparent throughout this thesis. They should be constantly kept in mind because they allow us to suggest the things we do. The next section presents a quick summary of the structure of this thesis.

## 1.5 THE STRUCTURE OF THIS THESIS

Regarding structure, the thesis is divided in three parts: introducing the problem, suggesting the solution, and a retrospective discussion of the results. The first part consists of chapters one and two; the second part comprises the main bulk of the thesis from chapter three through seven; and finally part three is chapter eight. A brief review of each chapter:

| Part | Ch. | Description |
|---|---|---|
| 1 | 1 | Introduction of the problem, focus of the thesis and research method used |
| | 2 | The problem in dept, the case company, and the problem seen from different perspectives |
| 2 | 3 | Rationale and the problem in different contexts, inspirational sources, and solution suggestion |
| | 4 | Theoretical walkthrough with focus on looking for useful knowledge snippets |
| | 5 | The models, system breakdown, encapsulation, and communication models |
| | 6 | Testing of the models |
| | 7 | Guideline for model making |
| 3 | 8 | Discussion of different aspects, answers to research questions, and conclusion |

The next chapter presents an in-dept look at the problem, some wonderings about the general characteristics of the problem, and how it could be solved.

# *Chapter 2*

# THE PROBLEM

This work is rooted in a problem recognized at the case company. The company is a world-leading manufacturer of pumps and pump systems which has been really successful in modularizing a part of its product assortment. They have utilized black boxes with multi functions that only are made into a specific variant when installed. This is done to achieve even larger economies of scale in production. This extreme postponement strategy has led to embedded software in each of the black boxes and a lot of parameters that decide what parts of the functionality are used each time. Hence, the task of giving variance to the system has been moved out of the production hall and downstream to installation at the customer site.

We use this case to induce a general solution suggestion that incorporates known techniques such as rule-based knowledge systems into the work process. To this end, we need to identify the main traits of the problem, why it exists and how to deal with it, and in this chapter we try to do just that. This chapter concludes with a solution suggestion that serves as an input to the next chapter, the literature search.

## 2.1 THE PROBLEM AS SYSTEMS OF PUMPS

In this chapter, we describe in detail the products used to induce the general problem. This description is an artefact view of the problem and tries to conceptualize a solution thereof. The induction to a more general problem comes later on.

The case in question is pumps and pumps joined together into a system of pumps. A specific product family was selected that was thought to give the best chance of inducing a general description. The criteria for selection were: the product must have great variance, the variance must partly be generated in software, and the number of elements in the systems may not be predetermined. The product that fitted best was the boosting pump, mainly used for fresh water, which is electronically controlled but can still be combined with other pumps to form boosting systems.

Both the pump as a system and a system of pumps could be used, but since this work is about knowledge structuring, it is preferred that all subsystems have embedded software, because this would facilitate the work and hopefully allow for a shorter path to induction. In this spirit, let us describe the products, first the individual pump and then the controller and censors. The pump is a centrifugal pump built up in a modular fashion. It has four main organs: power, control, coupling and flow as shown in Figure 6.

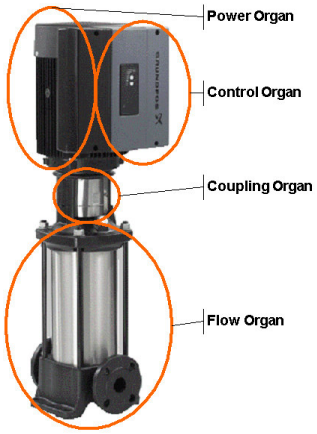



**Figure 6 – Grundfos electrically controlled pump and its main organs**

There are well-defined interfaces between the organs. Interfaces between power, control and coupling are standard and locked, meaning that one organ can be replace by different types of the same organ. The interface between coupling and flow is interlocked and hence not standard. It is not possible to change either the coupling or flow organ without it influencing the other organ, at least in some cases. Variance can be accomplished in all four organs. Hardware variance in the flow, coupling and power is not relevant in this work, but the effect on the control organ is. When some hardware is changed, the control organ might have to be adjusted. For example, if a larger motor is selected, some parameters in the controller have to be changed to match different currents, amperage, speeds and so on. The pump has a control organ and is hence a self-containing system, but it can also relinquish control to an external controller.

The controller is like the control organ in the pump but without the actuator, which is the frequency changer built into the pump control organ. The controller differs from the pump control organ in three main aspects: it can control more pumps, it has better user interface, and it has more applications. The controller is shown in Figure 7.


**Figure 7 - Grundfos controller**

The most interesting setup is when several pumps are joined together with a controller to form a booster system, as shown in Figure 8. A sharp reader will notice that the booster system (Figure 8) has the same organs as a single electronically controlled pump (Figure 6).

The former differs mostly from the later in its capacity to run complex applications. There are many more things to "play" with when there are many instances of each organ. One of the most interesting aspects is the cohort and coordination between the control organs. It is precisely this interplay that serves as the main input to this thesis.

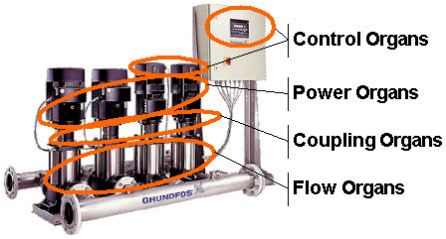



**Figure 8 - Grundfos booster system**

Now that the products used for this thesis have been shown, let us look in next section at higher abstractions of these same products and identify their processes of manufacturing and installation.

### 2.1.1 THE PROCESSES INVOLVED

Framing the problem is necessary, especially with regard to the process of manufacturing and installing the products. We do not examine manufacturing in general but only the aspects related to software parameters. To do this, we draw an abstract view of the product, based on the organs, and then identify the different stages the setting of parameters goes through. For example, a single E-pump goes through five stages in which software parameters are set or checked. The stages are: *Clean box*, *MGE motor*, *E-pump*, *Pump & censors* and finally, *Application*. The stages are shown in Figure 9.

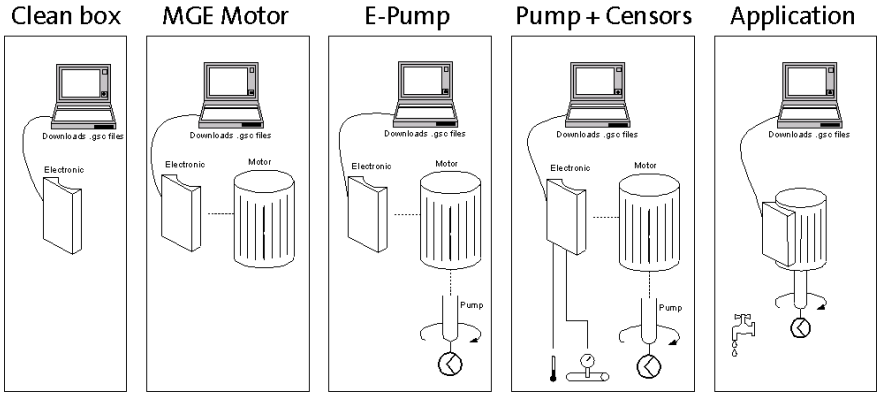



**Figure 9 - The process of assembling and commissioning a single pump**

A short description of each stage is as follows:

| | |
|---|---|
| Clean box: | Testing the sanity of print board in control organ |
| MGE Motor: | Mating of control organ to power organ. Setting of parameters related to eliminating functionality in control organ that power organ cannot perform. |
| E-Pump: | Mating of joined control / power organ to a joined coupling / flow organ. Setting of parameters related to eliminating functionality in control organ that power / coupling / flow organ joined cannot perform. |
| Pump & censors: | Mating censors to the E-pump. Setting of parameters related to eliminating functionality in control organ that E-pump cannot perform. |
| Application: | Selecting a single application that matches customer needs. |

From the descriptions, one can see that the first four stages entail narrowing the selections possible in subsequent stages, while the last is about selecting a single instance. The stages for a system solution are similar. Let us now look at an abstraction of a system of pumps.

The organs present in a pump system are the same as in a single pump, but the complexity increases as each organ now has many instances. This allows the system to solve the same application with different combinations, i.e. the same application can be achieved by using different combination of organs. An abstract picture of a pump system is shown in Figure 10.

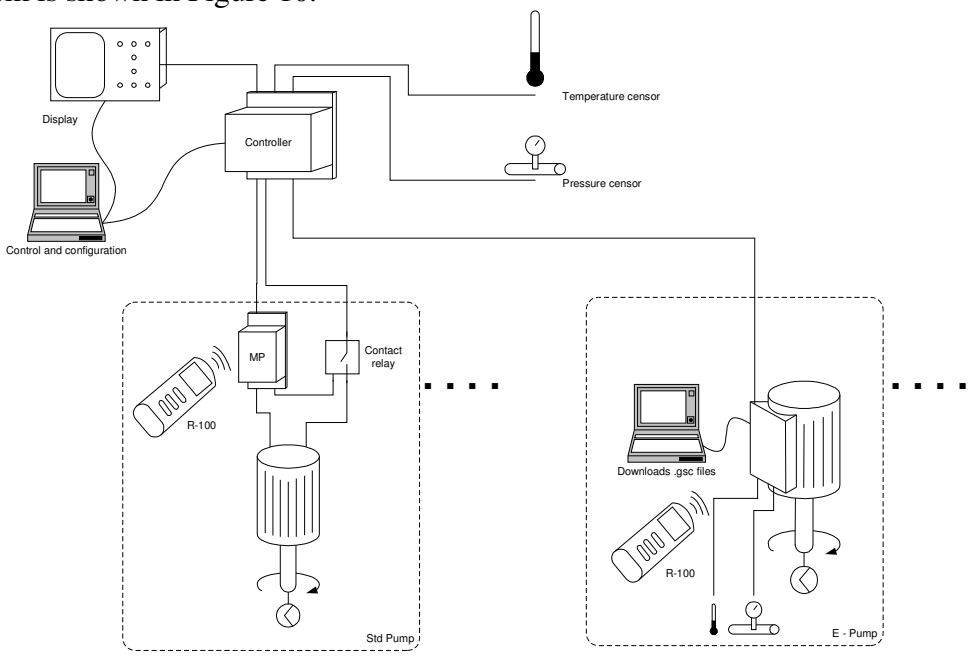



**Figure 10 - A pump system**

If we assume that the E-pumps arrived assembled at the customer site, then the pump system only goes through the last two stages of installation described above (see Figure 9 and following description).

To better put this in perspective, let us look at the evolution of a customer solution. It starts with a need of the customer and then goes through several stages to hopefully end in customer solution. It can then be altered afterwards to meet with new or extended demands or simply to be maintained. This lifecycle of a solution is shown in Figure 11.

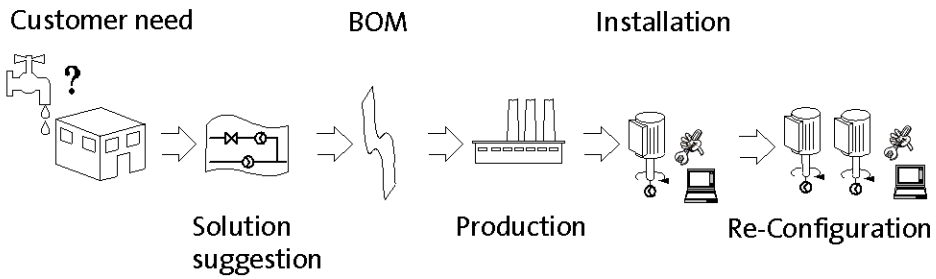



**Figure 11 - The life cycle of a pump solution**

These six stages of the solution lifecycle can help us to identify the flow of information and to see what is needed to complete the different tasks.
A description of the stages:

| | |
|---|---|
| Customer Need | This is where the end-customer realizes that a general need has to be fulfilled. It is usually on a high abstraction level like: "I need water to come from my tap, everywhere in the building and regardless how many are using water concurrently". |
| Solution suggestion | A need is translated into specifics. The formulated need translates into: "System has to maintain 2 bar pressure everywhere in the pipes". This and inquiry into the physical aspects of the customer site lead to a solution suggestion. The suggestion includes measurable specifics like capacity, fluid specifics, performances (like 2 bar) and operational parameters (including redundancy etc). |
| BOM | The solution suggestion is translated into a specific bill-of-material, e.g. a set of products selected to match the suggestion. |
| Production | The BOM is then produced, packed and shipped. |
| Installation | A set of crates (packages) arrives at the customer site and has to be installed to provide the solution to the end-customer's need. |
| Re-configuration | Maintenance, expansions and changes to the system carried out through the years. |

Both the five stages of installation and the six stages of solution lifecycle are used to describe the different kind of scenarios that arise when commissioning and maintaining complex product systems like the pump system mentioned above. Let us examine some trends worth mentioning in what has been described.

### 2.1.2 HARDWARE AND APPLICATIONS

Just by looking at the description of the assembly process (as shown in Figure 9), one can easily group the stages into two main activities or tasks: the matching of hardware and the selection of a single solution. The former is shown in Figure 12.

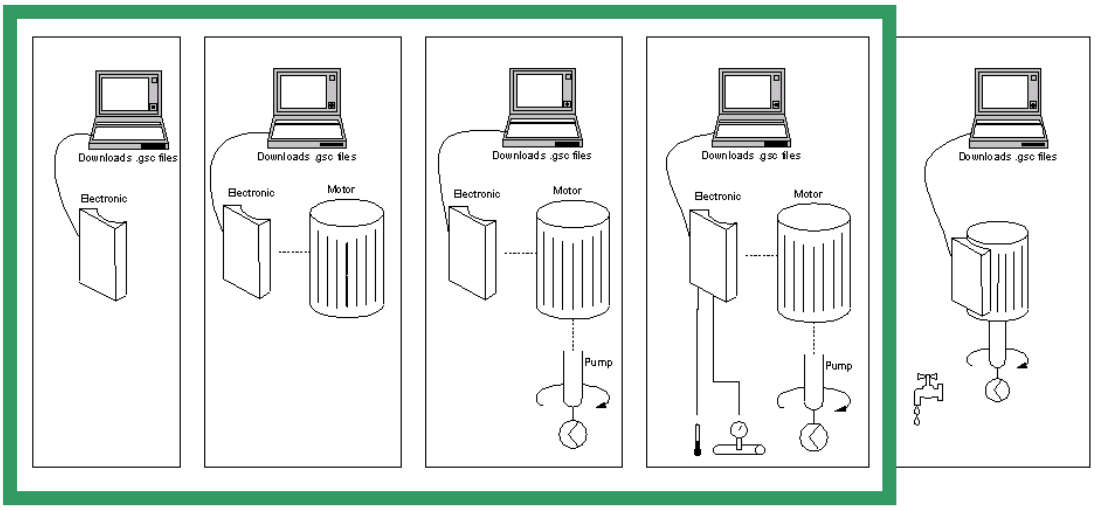



**Figure 12 - Matching hardware to form solution space**

All these stages (boxes in Figure 12) have to do with eliminating "illegal" functions from the system. They could also all be performed in a single stage. What

characterizes this action is that it does not rely on any outside information, given that the organs can communicate properly with each other. In other words, if the organs could tell each other about their inner workings, they could autonomously remove illegal functions. This can be seen in the solution lifecycle and is shown in Figure 13.

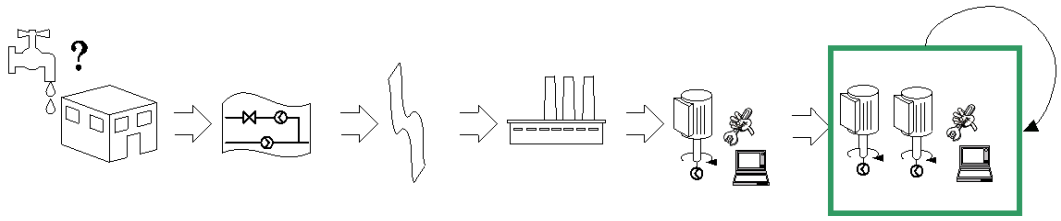


**Figure 13 - Adding hardware to existing solution requires mainly hardware matching, given external controller**

The other main task is the selection of an application. This is the last task in the stages of installations (Figure 14).

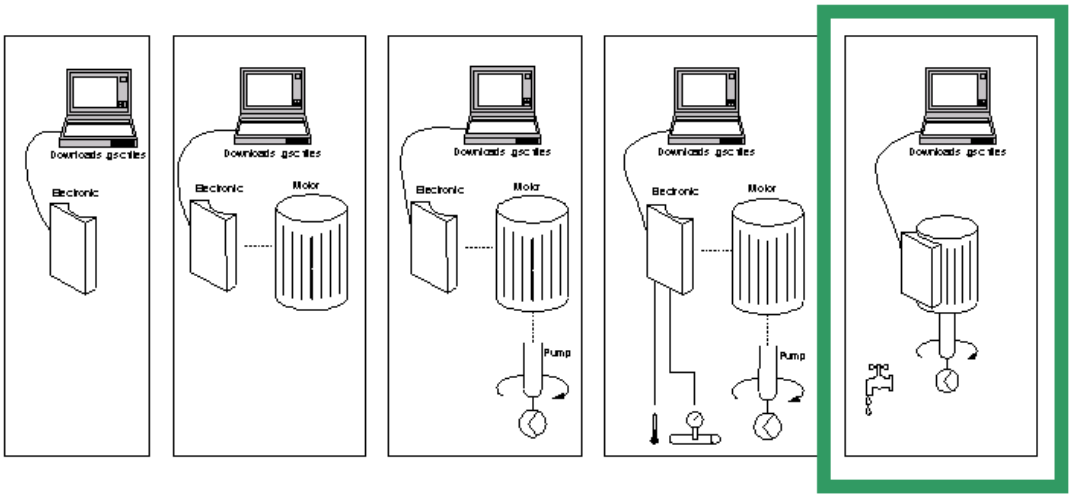



**Figure 14 - Selecting an application**

This task requires a lot of information, since it sets the hardware to solve the customer need. It thus requires information from all the other stages of the solution lifecycle, as shown in Figure 15.

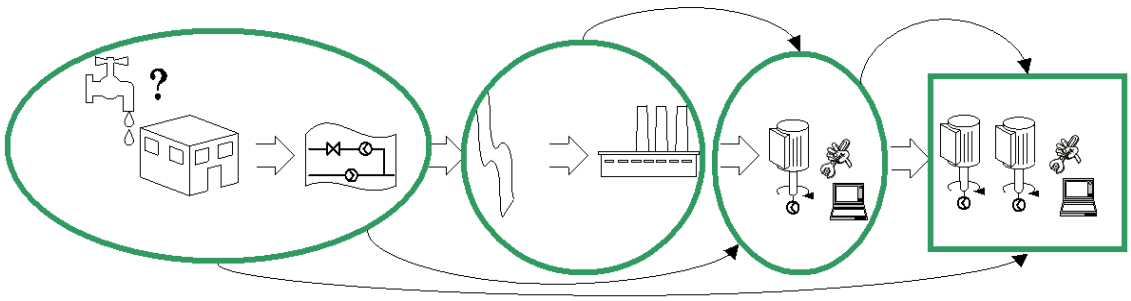


**Figure 15 - Selecting an application requires knowledge from all previous stages**

Information from the other stages can be both explicitly and implicitly stated. In practice, the rationale used in mapping customer need to a solution suggestion is often lost and is not accessible to the persons making the installation, who often have to deduce from the hardware present and the environmental context what application is relevant.

The installations that have to be made are not all equally complex. Let us look at some scenarios to identify general tendencies that may help in the induction to a general problem statement.

## 2.2 SCENARIOS TO SOLVE

While observing both the manufacturing and installation processes at the case company, several scenarios where described as stereotypes of the problem at hand. These six scenarios are described in turn and an attempt is made to apply the observations we have made until now.

The scenarios are now described in the order they appear, e.g. from manufacturing through installation to maintenance. In each, we try to identify the intake, who are the customers and who are the players (suppliers).

### 2.2.1 SCENARIO 1 – BUILDING AN E-PUMP

Manufacturing of the pump is the first scenario to look at. Here, the mating of hardware is dominant since the final customer is probably not known and therefore the application is not known either. This is shown in Figure 16.

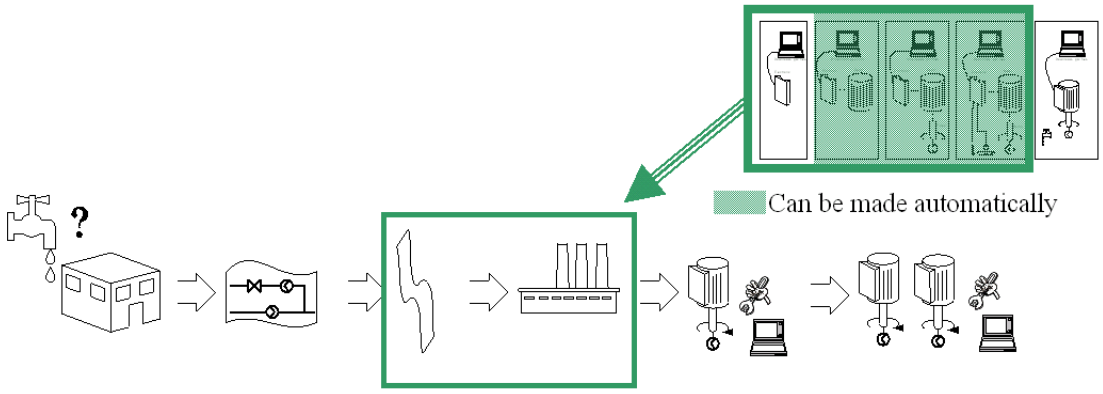



**Figure 16 – Scenario 1 – Building an E-pump**

In this scenario, only one of the two main tasks is present, the mating of hardware. This applies mostly to pumps, but also to pump systems that are completed in-house. They do differ a little, however, since they have a clear application, and therefore we leave them out. This process is well managed today with SAP and other tools in the case company.

### 2.2.2 SCENARIO 2 – COMMISSIONING THE SYSTEM

With the pumps produced, the next scenario of interest is the first-time commissioning of the system, the installation. Now we have a group of crates standing at the customer site, and they have to be installed to perform the task needed to support the customer. Here we have both of the main tasks, mating of hardware and selection of application. This is shown in Figure 17.

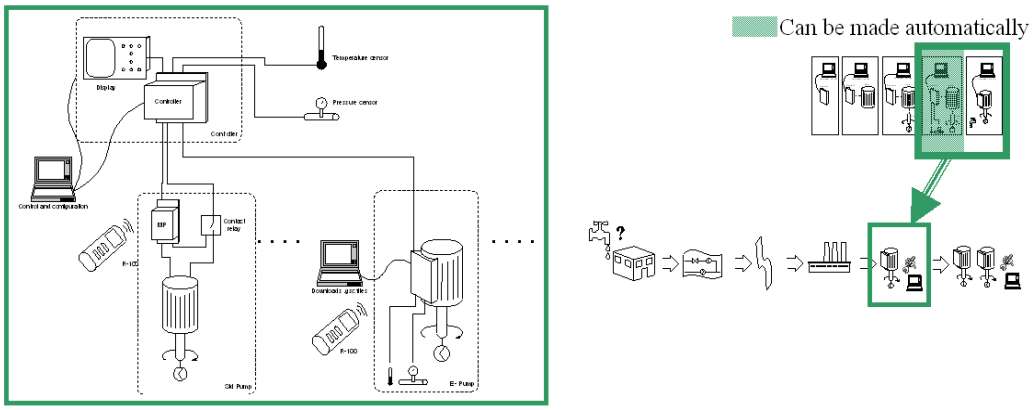



**Figure 17 - Scenario 2 – Installing the system**

In this scenario, the system has no previous knowledge and hence everything is needed from "outside". If hardware could communicate, this part could be solved automatically. This happens in the installation stage of the previous mentioned solution lifecycle (see Figure 11). After first-time installation, several other scenarios are possible in the re-configuration stage. There are three scenarios with single subsystem failure, which could also be called single organ failure. The last scenario in the re-configuration stage is making additions to the system.

### 2.2.3 SCENARIO 3 – PUMP CONTROL ORGAN

The first of the failure scenarios is the failure of the control organ in a single pump. Failure of other organs in a single pump do not require any setup changes as no computer is in them and hence no parameters. They are therefore ignored here. When a control organ goes, the system "loses" its application knowledge, and thus the new control organ cannot be supplied automatically. This is shown in Figure 18.

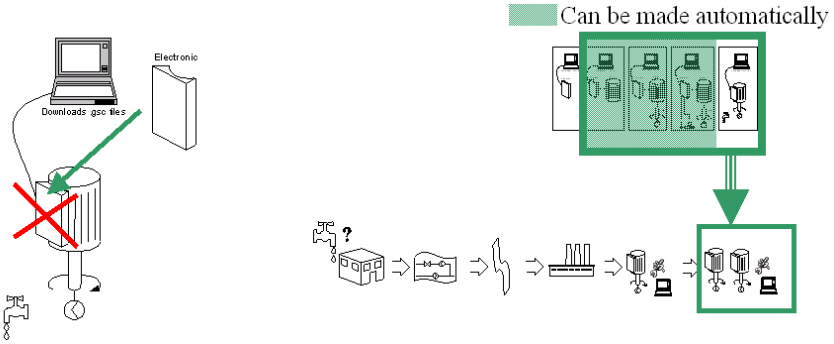



**Figure 18 - Scenario 3 – Replacing control organ in a single E-pump**

It is possible to make all the mating of hardware automatic, but some input will always be required to complete the application setup.

### 2.2.4 SCENARIO 4 – PUMP CONTROL IN A SYSTEM

A twist on the single E-pump scenario is when such failure occurs in a system of pumps. Now, it is possible for the system to "retain" the application information even though the control organ in the pump fails. The "knowledge storage" can take place in the main control organ or be supplied from the other pumps in the system. This is shown in Figure 19.

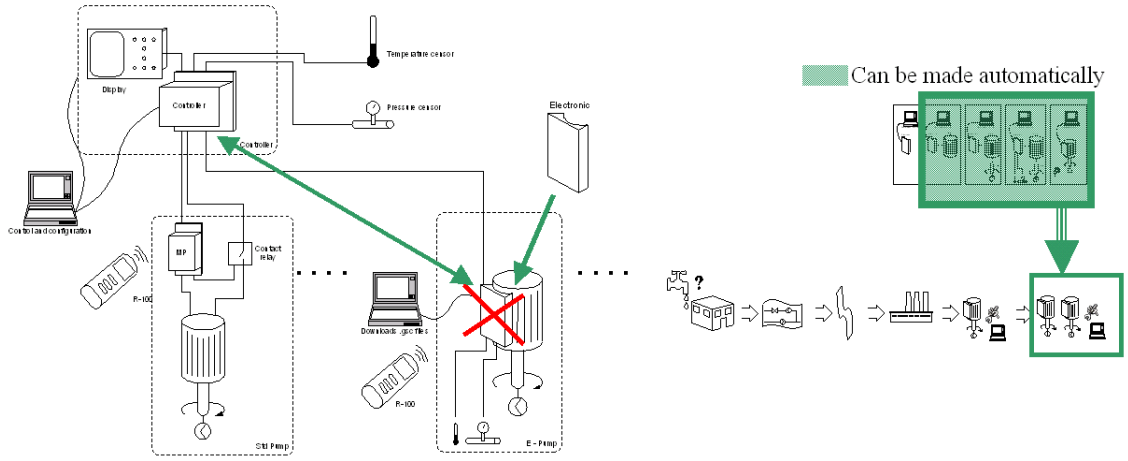



**Figure 19 - Scenario 4 – Replacing control organ of a pump in a pump system**

As a consequence, the replacement only requires a hardware switch; all parameters would be set automatically. This would not be the case if the control organ were in the controller. Now, let us look at failure of the control organ in a main controller.

### 2.2.5 SCENARIO 5 – SYSTEM CONTROL ORGAN

Main controller failure would result in application knowledge lost in the system. This is because we assume that this organ has greater capacity and that its knowledge cannot be stored in the other control organs in the system. This would require that application information had to be re-entered. This scenario is shown in Figure 20.

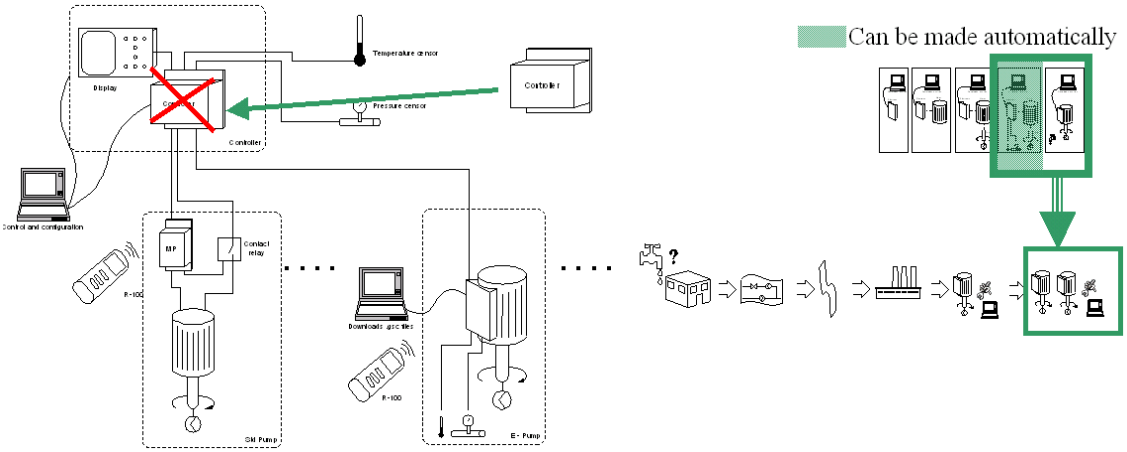



**Figure 20 - Scenario 5 – Replacing control organ in a pump system**

The last scenario is the extension of the system.

### 2.2.6 SCENARIO 6 – ADDING TO SYSTEM

After a system has been installed, it can be augmented. In such cases, the knowledge resides within the system and all aspects to the configuration can be done automatically, as shown in Figure 21. The six scenarios presented here are based on the tasks identified in the case company. An observant reader would have seen by now that the trend is toward merging these scenarios.

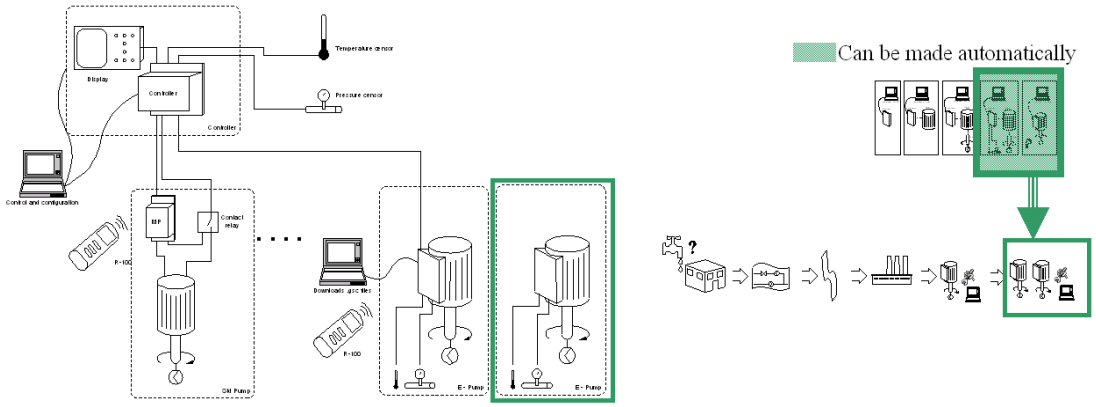



**Figure 21 - Scenario 6 – Adding to the system**

It is possible to generalize these into arch-type scenarios, which we do in the next section.

### 2.2.7 ARCHTYPE SCENARIOS

The previous six scenarios can be described by two archetype scenarios. The first archetype scenario is where application knowledge is retained in case of change, as in scenarios one, four and six. The other is where the application knowledge is lost, as in scenarios two, three and five. Therefore, the suggested solution has to support both of these. In the first archetype scenario, it is worth noticing that sharing of knowledge becomes crucial when new sub-systems are mounted.

## 2.3 UNDERSTANDING THE PROBLEM

Now, we look at the problem from different viewpoints and try to decompose and categorize it. Along the way, we identify general concepts that are of use later on. We describe these concepts where it is appropriate, and in the next section summarize our findings in the form of definitions. This section deals with context, people, knowledge acquisitions and techniques that may be helpful. This section should be understood as a pondering on how a problem can best be solved. In the next chapter, we then use our findings here to browse through the many different domains for solution snippets.

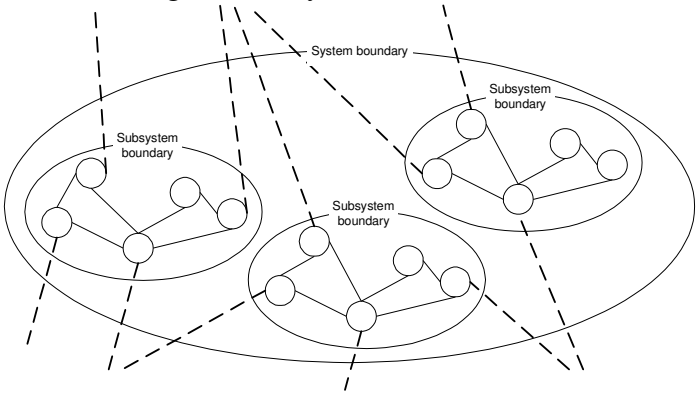



**Figure 22 - Subsystems combined to form a system**

The concept of embedding a configuration engine into each subsystem and the benefits of this can be explained by system theory, here presented graphically with elements, relations and boundaries, as for example in Skyttner (Skyttner 2001), but built on von Bertalanffy's ideas (Bertalanffy 1969). Let us think of a system (Figure 22) as having elements that have internal relations.

Some of the elements require input from the environment (e.g. users) and therefore transcend the system boundary. Here, it is assumed that the elements are some kind of parameters that have to be set. Let us define information flow across the system boundary.

**Definition I: Decision Variables**

A variable that breaks the overall system boundary shall be called *Decision Variable* (DV). These are variables that need the user (here, both persons and computers) to make a decision. If decision variable only breaches the subsystem boundary and can be set internally by the system, it is called *Internal DV*. A decision variable that breaks the overall system boundary but is only active when another DV has a specific value is called *Tentative DV*.

Some DVs are set with internal relations, while others require input from outside the system. Note that the abstract figures of the system that follow do not state anything about what the system does; only that it needs inputs to be set to a working state. For explanatory purposes, let us assume that the system is a multipurpose system, and the user inputs needed are parameters to select some of the intended behaviours. If many of such systems are combined to a larger system (Figure 23), an "installation" problem arises, where subsystems have to be "matched" or configured to the overall system-intended application. The following rationale only deals with the problem of installing such systems, but not the subsequent use or purpose (this can also be called system assembly). It is about making a system that works, but not about what it is to do.

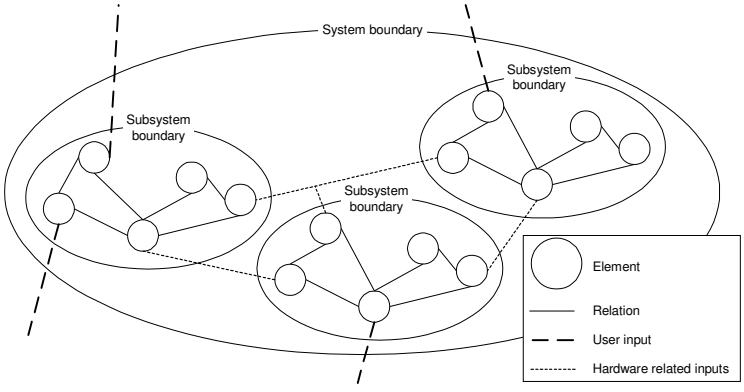



**Figure 23 - Some relation are hardware-setup related**

Back to Figure 22 and how the various user inputs are related: Someone, like the user or an external information system, has to set all the user input parameters to make the system work. But the inputs are not independent; some of them can be related in one of two ways: 1) they are to match the hardware together, i.e. one subsystem is connected to another, and the parameters make each sub-system "aware" of the others; or 2) they are related in the use of the final system, the application, i.e. when the overall application is decided, some of the inputs connect. Both of these relations can easily be expressed with rules, constraints or mappings in an integrated system. Things get

trickier when the final system is constructed of unknown subsystems, i.e. when it is not predefined which subsystems are present in the final system. The enclosing of the solution space or the matching of hardware is the first thing that relates some user inputs to others. Graphically, this can be shown by connecting interrelated inputs, so when one input is set, some others will automatically also be set, as shown in Figure 23

The interrelating of these inputs is parallel to constraining the solution space. As these are completely hardware-related, we name them *Hardware-induced configuration*. That gives us another definition:

**Definition II: Hardware-Induced-Configuration (HIC)**

| |
|---|
| A variable that has to do with connecting hardware together is called *Hardware-induced configuration*. An important aspect of HIC is that if information is stored internally in the system, it can be set completely automatically. |

The nature of these relations, and the fact that all needed information for connecting the subsystems lies within the overall system, supports the argument that this linking should be accomplished completely automatically. The other kind of interrelation between inputs is the application-related inputs. These can be set when the overall usage of the system is determined. Once the application is selected, some inputs will relate to each other, and by answering or setting one input, several others may be determined. This can be shown graphically just like the hardware setup, as in Figure 24.

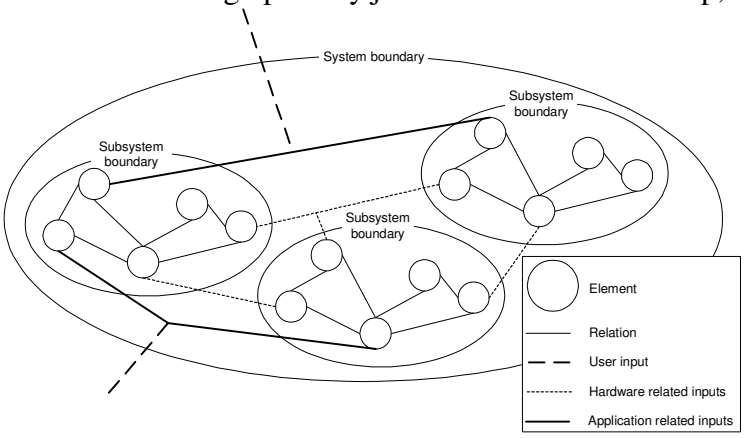



**Figure 24 - Some relations are application-related**

Application relations rely on user inputs to be set, so they are most likely not automated, but the relationships should be identified before the parameters are set. This relates to the selection of a specific solution and is named *Application-induced configuration.* What may not be stated explicitly here is that part of the configuration can take place inside each subsystem and that it is completely "contained", i.e. each subsystem will make sure that its internal workings are in order and no illegal settings are present.

**Definition III: Application-Induced-Configuration (AIC)**

| |
|---|
| A variable that has to do with selecting a single application or a specific solution is named *Application-Induced Configuration.* Contrary to HIC, AIC always relies on user inputs to be set; thus, they are most likely not automated. |

An example illustrates this:

> Think about your home and the television and DVD player that most of us have. When installing the DVD player, it is necessary to use the menu to tell the player what kind of TV is present, i.e. normal or widescreen. This is in essence a hardware-induced configuration. Once a DVD disk is placed in the player, it is necessary to select an "application", e.g. subtitles or a language. This is the selection of a specific solution, or application-induced configuration.

This example could of course be made much more complex, but it should highlight the two important aspects of embedded configuration, i.e. the hardware-induced and application-induced configurations.

Combining subsystems to form a system in which the subsystems have to be configured to work in the overall system is a tedious task. As seen in Figure 22 to Figure 24, it is possible to reduce the user inputs by relating inputs to each other. This encapsulation of both hardware awareness and application hopefully require fewer inputs by the user. By connecting complexity to the number of activities, it can be argued that complexity is reduced in such a concept, and hence the usability for the user is increased. This, of course, remains to be seen (and proved), but the working hypothesis is that this holds true.

It is easy to put forth such concepts but to show how to make them viable and construct them is another matter. When constructing such a system, several issues have to be clear. The setup has to be tolerant towards new subsystem versions and be able to demonstrate redundancy by reconfiguring if some subsystems should fail or partially stop functioning. This requires that the product knowledge of each subsystem has to be stored internally in each subsystem and must then be able to communicate meaningfully to cope with the overall system. The reasoning in this chapter also indicates why communication between modules should be the focal point of the concept. To be able to inter-relate hardware and application settings between subsystems, it is necessary to know what each subsystem requires in order to function and how to ensure that such knowledge sharing deals with the issues mentioned earlier.

With this rough split (between hardware and application) in mind, let us look at the problem in detail and see if some further decomposition or characterization can be suggested.

### 2.3.1 PROBLEM CHARACTERIZATION

After staying with the case company for several months and observing their work and processes and analysing their products and product systems, the researchers were able to suggest the following problem breakdown. Using Figure 25 as a map for the different phases that an installation goes through can give us a good breakdown of the problem.

Lets first see what is in each phase and then see what models are needed to successfully describe the knowledge needed in each phase. As seen in the visualization of the whole problem in Figure 25, several phases are involved.

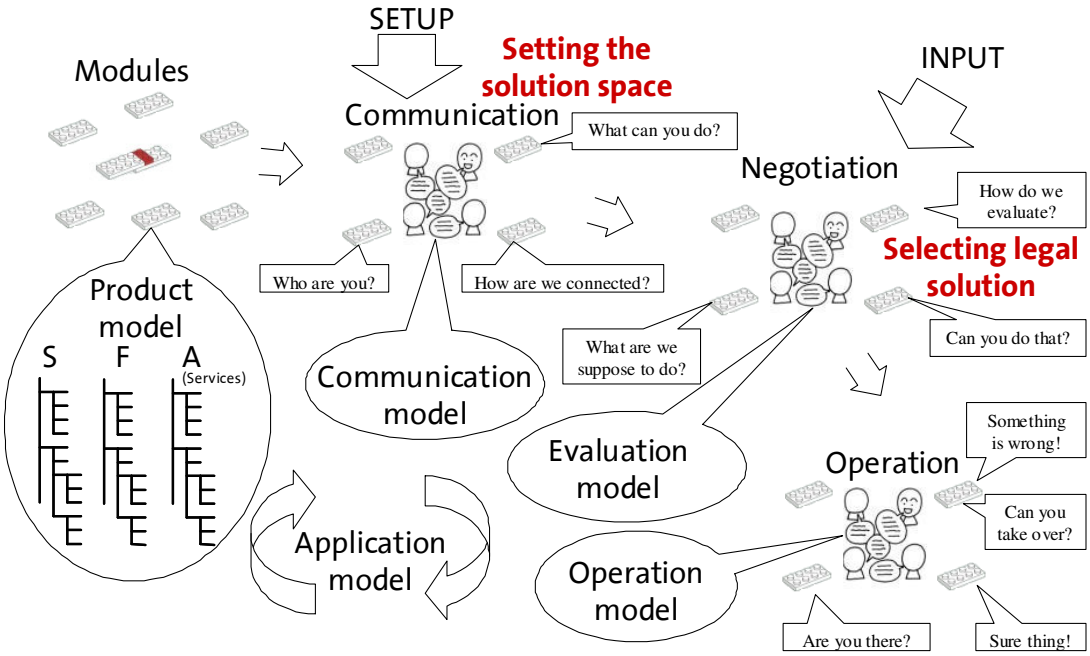



**Figure 25 - Phases of system setup and their models**

These phases make for a relatively easy and clear characterization of the problem.

**Table 1 - Models in phases of system setup**

| Models: | Description | Catagory / Main character |
|---|---|---|
| Product Model | Description of product data, structure, functions, behaviour and how theses can form services to offer others | Information structuring on different abstract levels<br><br>**=> Acquisition** |
| Communication model | A battle plan for what knowledge is needed to successfully communicate and set the system boundary | Informing all subsystems and constructing a single solution space<br>**=> Communication** |
| Evaluation model | Once solution space is set, one setup / configuration / solution has to be selected. This selection is based on some input and evaluation. This model is rather unclear at the moment | Evaluating best solution on basis of input<br><br>**=> Optimization** |
| Operation model | Mostly "hardcore" logic that controls, measurements used to control behaviour, but also some redundancy and protection control | Control, fault detection and protection of system<br><br>**=> System Operation** |
| Application model | Connects all the models and ties them to customer input, i.e. what function is used, how communication is done and what operation issues are relevant for each application | Across-department acquisition of information<br><br>**=> Accumulation of corporate knowledge** |

Four phases are identified: *Modules,* where the system is unassembled, *Communication,* where the hardware connections are made, *Negotiation,* where the application is negotiated or selected, and finally *Operation,* where the system is commissioned and in operation. The communication phase is about defining the solution space, and the negotiation phase is about selecting a single solution. Coupled to these phases are five models. A description and characterization of the model is shown in Table 1.

These models were suggestions made early in the process. Since it later turned out that they are not equally important, some refinements were made. The final model group is described in Chapter 5, page 109. At this stage, these models serve as categorization and will aid in suggesting ways to deal with the problem.

An important thing to look at is the people involved, and what skills are required for the work, both in regard to the current problem solving at the case company (the operation) and in regard to designing / redesigning the products. Let us start with the cognition involved.

### 2.3.2 COGNITION INVOLVED

The product system in question is operated and designed by people. These same people have some skills and use some cognition to carry out their work. Let us evaluate what is needed to fulfil two kinds of tasks: operation of each model, and the design of the same, as shown in Table 2. Let us use Bloom's taxonomy of cognitive objectives (Bloom 1956) for the evaluation.

**Table 2 - Cognition needed in each model**

| Model: | Operator | Designer (model maker) |
|---|---|---|
| Product model | Knowledge, Comprehension, Application (KCAp) | Knowledge, Comprehension, Analysis, Synthesis, Evaluation (KCASE) |
| Communication model | Comprehension, Application (CAp) | KCASE |
| Evaluation model | Application, Analysis, Evaluation (ApAE) | KCASE |
| Operation model | Application (Ap) | KCASE |
| Application model | Comprehension, Application, Analysis, Synthesis (CApAS) | KCASE |

This is an attempt to evaluate the models in question and see how complex they will be. Although Bloom's taxonomy is for cognitive learning, it should give insight into what abilities people need for working with each model. There are distinct differences in abilities between operators and the designers that make the models. This talk of cognition might seem a little out of place here, but we return to it in the next chapter, where we argue for our solution.

### 2.3.3 CATEGORIES OF KNOWLEDGE

Another way of evaluating the model is to look at the knowledge needed to populate them. For this purpose, we use categories of knowledge suggested by Professor Staffan Sunnersjö (Sunnersjö 2006) in the phd course, *Intelligent computer systems for automated engineering design,* held in Jönköping, Sweden, in the spring of 2006. Connecting these categories to the models from each phase results in the following Table 3.

This would be the combined knowledge of the operators and the designer. Since the models are only a suggestion at the moment, this is only an educated guess as to what is relevant and should be seen more as a roadmap for where to look when the construction of the models is underway.

**Table 3 - Categories of knowledge**

| Little use ◔<br>Some use ◐<br>Relevant ◕<br>Very relevant ● | Product model: Artefact model | Product model: Functional model | Product model: Services | Communication model | Evaluation model | Operation model | Application model |
|---|---|---|---|---|---|---|---|
| Purely empirical knowledge | ● | ◐ | ◐ | ◕ | ● | ◕ | ● |
| Rules of thumbs (heuristics) | ● | ● | | ◕ | ● | ◐ | ● |
| Praxis as knowledge (heuristics) | | | ● | ◕ | ◕ | | ● |
| Analogy | | | | ◕ | ◕ | ◐ | ◕ |
| Common sense | | ◐ | ◐ | ◕ | ● | | ● |
| Logic | | ● | ● | ◕ | ● | ● | ● |
| Geometrical (spatial) knowledge | ● | | ◐ | ◐ | ◐ | | ● |
| Numerical algorithms and mathematical formulas | ● | ◐ | ◐ | | ◐ | ● | |

### 2.3.4 RELEVANT TECHNIQUE IN EACH PHASE

Formulation of the knowledge needed for each phase is very important. The researchers also think it is very important that the final models are easy to use, intuitive, give a good overview, can be made in different detail levels, and focus on minimizing information flow – i.e. information flow both in respect to modules but also between models. This should aid in implementing a computerized solution, as models should support modularization of the task.

Automation of the process requires that each model is made explicit in such a way that it can later be supported with computers. The following is a suggestion for a set of methods / tools that can hopefully help bring the different models to life (see Table 4). The methods listed here are drawn from course material (Sunnersjö 2006). More tools are most likely needed to identify, map and construct all the models.

The degree of automation of the models differs. When this work was done, the researchers predicted that several models should and most likely could be made completely automatic. These would be the product model, communication model and operation model. The automation would be in regard to the operators, not the designer. The reason for their full automation is that all relevant information is present and can be usefully structured to support it. The only input needed in these three models is the "setup" arrow from Figure 25, which is connection of hardware, perhaps with cables, some kind of plug-ins and so on. But after hardware connections were made, the configuration would be automatic. Later in the process, the operational model was put on hold, but we discuss this in Chapter 5.

**Table 4 - Knowledge-based techniques**

| Might do ◔ / Worth checking ◐ / Relevant ◕ / Very relevant ● | Artefact model | Functional model | Services | Communication model | Evaluation model | Operation model | Application model |
|---|---|---|---|---|---|---|---|
| | Product model | | | | | | |
| Product Variant Master (PVM) | ● | ● | ◐ | | | ◐ | ◐ |
| DSM | ● | ● | ◐ | ◐ | | ◕ | ◕ |
| IDEF0 | | ◕ | | ◐ | | | |
| MFM / FBS / GTST / GFM | | ● | | ◐ | ◐ | | |
| Functional Basis | | ● | | | | | |
| UML / MML (MOKA) | | ◕ | ◕ | ● | ◐ | ● | ● |
| Petri nets | | | | ◐ | ◐ | ◐ | |
| Semantic nets | | | | ◐ | ◐ | ◐ | |
| Workflow (BPMN) | | | | ◐ | | ◐ | |
| KQML / KIF / DAML | | | ◕ | ● | | | |
| Game theory | | | | ◐ | ◐ | | |

The remaining two models, the evaluation and application models, however, cannot be made fully automatic, at least not according to our knowledge of such systems at the current time. These models are tightly connected, as they depend a lot on customer's requirements and the context within which the system is going to be used. The input needed (the "input" arrow from Figure 25) will always have some manual element, since the system as a whole can probably never have all the information it needs incorporated. This input can of course be minimized by including much of the relevant data in the models and then encapsulating them into the system. It is the construction of the two last mentioned models that is going to be hard, both the general layout of the models but also the population of them with case-specific data. If the models can be made to a satisfactory degree, the next step can take place, i.e. putting them into a computer.

### 2.3.5 COMPUTERIZATION IN EACH PHASE

Based on earlier identifications of knowledge types and the tools that could aid in constructing the relevant models, a picture emerges of how to implement the models. It becomes quite clear straight away that no single solution type can solve all models. In going through the model suggestions, a spectrum of computerized solutions emerges that have to be melded together to form a coherent solution for all the models.

A trend emerges here; the product model(s) can most likely be computerized with relatively standard tools, while the other models will require more complex AI oriented tools. This also mirrors the type of knowledge included and methods used in each model. This also means that most likely there are no standard systems that can deal with all models, and a complete solution would require some original design and programming.

**Table 5 - Computerization of models**

| | Product model | | | | | | |
|---|---|---|---|---|---|---|---|
| | Artefact model | Functional model | Services | Communication model | Evaluation model | Operation model | Application model |
| Might do ◔ / Worth checking ◐ / Relevant ◕ / Very relevant ● | | | | | | | |
| Passive knowledge systems | ◕ | ◕ | ◕ | | | | |
| Executable knowledge systems | ● | ● | ● | | | ● | |
| Workflow systems | | | | ◔ | | | ● |
| Configuration system | ● | ● | ● | ◐ | | ◐ | ◕ |
| Evolutionary algorithms | | | | | | ◐ | ◔ |
| Case-based reasoning (CBR) | | | | ◕ | ● | ◐ | ◐ |
| Neural networks | | | | | ● | | ◕ |
| Intelligent agents | | | ◕ | ● | ◕ | ◕ | |

### 2.3.6 SOME RESULTS ON UNDERSTANDING

Using the techniques mentioned in the last couple of sections to characterize the different aspects of a project is no doubt beneficial but not without drawbacks; however, the positive effects far outweigh the negative ones.

The strength of using these techniques is that it forces different views and aspects on a project, which then extend the horizon and inspire new ways of thinking. This helps clarify the problem at hand and provides tools to aid in grouping. Problem clarification leads the way toward an action plan, a work plan for how to tackle the project. Another fine aspect of this exercise is to see the interplay between the automated process techniques and their computerization against the softer knowledge structuring and the cognition types needed. This task's greatest reward is the increased focus on what to do that arises from the improved transparency of the problem. This transparency is synonymous with an overview of the whole problem, a map that allows for sharper focus.

As with all methods and tools, there is a downside to this approach. The granularity of characterization is often quite coarse, which is especially apparent in knowledge and cognition grouping. The result is that differences between activities are not very great. This does not render the grouping useless, but it requires very careful contemplation of the groupings and their subsequent interpretation. The question rose along the way of whether cognition grouping aids in the work (at least with Bloom), since each phase is very much alike, the only significant differences being between operators and designer. The exercise requires a lot of "customization" of characterization in order to be useful in a project, and therefore it requires a time-consuming search for the right items to be included in the groupings. This also often means that it is hard to place the project into the categories, as that almost implies that the work is done and all things known. This concludes the analysis of the problem and some closely relevant aspects. From these wonderings, researchers are able to point out tracks to follow and where to search for a solution. And that is precisely the content of the next chapter.

# *Chapter 3*

# A RATIONALE FOR A SOLUTION

Having an increased understanding of the problem and its context, we can now begin to formulate a rationale for a solution. But first, let us ponder on the problem once more, now from different aspects such as business, production, organization and people. This can hopefully place the problem in an even larger context and point to domains where parts of the solution can be sought. Thus, this chapter is about the bigger picture, where inspiration can be sought and why. We conclude this chapter with a solution suggestion and an assessment of its impact. We start with different aspects of the problem.

## 3.1 ASPECTS OF THE PROBLEM

The problem identified and described in the previous chapter can be seen from different viewpoints. For an enhanced understanding of the problem and to place it in different domains, let us look at four views that can be said to either explain the problem or support a rationale for a solution. The viewpoints covered in this section are those of business, production, organization and people. The business viewpoint is about the pressure from the customer and business to increase variety but to maintain efficient operations. The production viewpoint is the operation domain's wishful thinking about staying in the nice and cosy environment of mass production where everything is "known" and its problems adapting to flexible manufacturing. The organizational viewpoint is about how organizations are not completely aligned with the strategy, operational efficiency or customers needs and how organizational structures are to be understood to make progress. The last viewpoint is about humans, people like you and me, our strength and weaknesses and how these should be taken into consideration. The last view deals especially with the limitations of our capabilities. In conclusion, a summary of all viewpoints is made and the question of why a solution should be devised is addressed.

### 3.1.1 BUSINESS VIEW

This section is about the business view of the problem – how increased customer focus has led to increasing product variety, which again has become headache for the organization. A fine starting point for this discussion is mass customization, as it captures precisely the points made above. Definition of mass customization, like that of Joseph Pine II, helps identify mass customization's implication for after-sales service.

> *... practitioners of mass customization share the goal of developing, producing, marketing, and delivering affordable goods and services with enough variety and customization that*

> nearly everyone finds exactly what they want.
> (*p. 44) in (Pine II 1993)*

Things like "enough variety", "affordable goods and service" and "exactly" point to great product variety and matching service, which has to be supported in the after-sale. Basic building blocks for mass customization are customer focus or customer needs (Piller 2005) and some combination of postponement (van Hoek 2001) and modularity (Fixson 2003), i.e. either one or both. Several authors, like (Duray et al. 2000) and (Salvador et al. 2002), have handled the aspects of modularity in mass customization. Others tie together modularity and postponement in mass customization implementations (Mikkola & Skjott-Larsen 2004). The impact of modularity (its different types) and postponements on after-sales service is unclear, but there is very likely a connection to the mass customization strategy taken.

Since not all mass customization strategies impact after-sales service equally, it is necessary to identifying the relevant factors and find out how they influence after-sales services. To do this, a quick review of some authors' views on mass customizations strategies is needed. Although mass customization was discussed somewhat before 1993, Pine's book on the subject is a good starting point. Pine listed five strategies (Pine II 1993) to implement mass customization. Here, he blends tangible product and service in different quantities at different customers' decoupling points. These are developed further in Gilmore and Pine's work, where they introduce the four faces of mass customization, or the Collaborative, Adaptive, Cosmetic, Transparent strategies (Gilmore & Pine 1997). Customers' decoupling point is the main intake in Lampel & Mintzberg's continuum of strategies (Lampel & Mintzberg 1996) where they draw a continuum from pure standardization to pure customization. A marketing view on mass customization (Kotler 1989) is formulated by Kotler but is not so relevant for our discussion here. More interesting is the view formulated by Piller, with his three dimensions for mass customization, or the style, fit and comfort and functionality approaches (Piller 2005). Another relevant approach is presented by Duray et al., who talk about different types of implementations of mass customization, or the Fabricators, Involvers, Modularizers and Assemblers (Duray, Ward, Milligan, & Berry 2000). All these strategies represent an axis in a multidimensional strategy space where the strategies do not map one-to-one between each other but rather many-to-many, maybe with some restrictions. Thus, two companies can belong to the same strategy in relation to one other, but belong to different strategies in relation to a third; an example could be an adaptive strategy that can be achieved with both modularizer and assembler strategies. Mass customization readiness (Da Silveira et al. 2001) with six generic success factors can be measured to help identify the appropriate strategy. Mass customization strategies to meet the customer needs impact the after-sales service provided.

In this thesis, it is the postponement and not so much modularization that is of interest, since this has been observe to cause added work in after-sales services. After-sale service can be said to consist of six activities (Wilson et al. 1999): Installation, Training, Routine Maintenance, Emergency Repair, Parts Supply and Software Services; and in addition customer risk reduction activities like service contracts and warranties (Ives & Vitale 1988), and the processes of supplying aforementioned activities.

| Design-related | Risk reduction | Service support-related |
|---|---|---|
| 1. Increase product reliability<br>2. Modular construction<br>3. Building in redundancy | 1. Warranties<br>2. Service contracts | 1. Improving service responsiveness<br>2. Reducing equipment repair time |

**Figure 26 - Ways to improve after-sales services (Ives & Vitale 1988)**

Connecting the six after-sales activities with the ways to improve these services can be depicted as in Figure 27.

| Risk reducing activities and the supporting process | Warranties / Service contracts | | | | | |
|---|---|---|---|---|---|---|
| Activities that are required to solve customer needs in after-sales service and their supporting processes | Installation | Training | Routine maintenance | Emergency repair | Parts supply | Software service |
| Product design | Physical product | | | | | |

**Figure 27 - After-sales service and its components**

The six activities that the customer must care for are all dependant on the actual product design. The ease of installing, training and maintaining (both routine and emergency) depends greatly on the physical product and its design. Overall, the industry is more interested in after-sales services than ever before. Some of the reasons are as follows: The time frame of after-sales, also known as the service life cycle (Potts 1988) or middle-of-life cycle, is much longer than for initial sales. The customers do not pay much notice to the cost of after-sales services, resulting in good profits on after-sales. The cost of selling to a new customer is much higher than the cost of selling to an existing customer. There is no or less competition, when it comes to selling after-sales services. Broadly speaking, it is easier to generate profit in the service life cycle than from the initial sale of a product, or as Wise and Baumgartner put it:

> *Clearly, in manufacturing today, the real money lies downstream, not in the production function.*
> *(p.134) in (Wise & Baumgartner 1999)*

The current view on how to benefit from the after-market is process-oriented, like the view offered by (Cohen et al. 2006), who suggest a different supply chain for after-sales service. This is a simple view with no pressure on radical product changes or service restructuring. Even though the literature points to opportunities for making money in after-sales services, this is not always the case. Sometimes a lack of knowledge about the services offered and their cost results in no money being made downstream.

To recapitulate the issues stated in this section: Mass customization to meet customer needs leads to

- increased variance, which demands
- flexible manufacturing with postponement and modularity, which
- impacts after-sales service heavily and increases complexity there.

Next is the production view, which has to play "catch-up" to the changing business.

### 3.1.2 PRODUCTION VIEW

When managing a project, the leader must know two things (preferably): where to go and how to get there. Literature on service (and on mass customization as well, to some extent) has rather been focused more on where to go and less on how to get there. The contribution here is aimed at the "how" part, and is structured to serve the "where" part already established. Connecting these two aspects requires a step back in order to identify the core of things. To illustrate this approach, think about the following: Optimizing the staff and queuing in a bank versus Internet banking. The former consists of optimization of current structure, while the latter is a completely rethought solution to the act of banking. It is the purpose of this work to move away from the optimizing view and engage in creating a solution for the act itself, or as Hammer once said:

> *Don't Automate, Obliterate.* *(p.104) in (Hammer 1990)*

We believe the way forward has two major components, smart products and modularized service. Allmendinger & Lombreglia state that without changing the products after-sales service, it will not reach its full potential (Allmendinger & Lombreglia 2005). It is also our conviction that if services are not defined properly, with modules and mapped to the increasing ocean of product variance the after-sales service will neither reach its full potential. Increasing variety of durable goods and matching service will most likely lead to increased complexity in after-sales services, both in regards to the goods itself and the making of service products to support them. To aid in solving this we like to give some rationale for how a solution to this problem could be like. The mantra here is like so beautifully stated by Takeuchi and Quelch in their customer service program.

> *Be efficient first, nice second.*
> *(p.144) in (Takeuchi & Quelch 1983)*

The aim is to make the problem explicit, so effective and efficient solutions can be constructed. Variety also forces operation management to react in order to be able to produce at a reasonable cost level. Instead of using stocks and forecasting methods (Chambers & Mullick 1971) to estimate demands, it is necessary to evaluate the order penetration point (Olhager 2003) in the supply chain and to apply most likely both IT and Business Process Reengineering (BPR) from industrial engineering (Davenport & Short 1990) to address the problem. This will probably lead to some operation innovation (Hammer 2004) that could aid in great variety production. There are two key components observed in tackling this, postponement and modularity (Mikkola & Skjott-Larsen 2004). Let us examine these a little closer.

Postponement is about delayed product differentiation and can be done both from a design and manufacturing perspective (He et al. 1998). Postponement was well known in the heyday of mass production, but was lost somewhat with all the talk of flexibility and agility in the eighties and nineties. The rediscovery of postponement (van Hoek 2001) has since emerged with increased modularity. How to choose the right strategy for postponement (Pagh & Cooper 1998) relies on many factors and form postponement (Forza et al. 2005) is a one possible type. At the case company,

postponement has been taken to extremes by introducing embedded software (Lewis 2001) in the product and first establishing the variance at installation. It is precisely this operation strategy that has led to the problem investigated here.

A recap of the production view includes the following points:

- Increase variety has been solved with postponement and modularity
- Too much focus on optimizing and less on obliterating
- Smart products have a dual purpose: to maintain mass production efficiency and fulfil a wide spectrum of customer needs
- "Be efficient first, nice second" is just as valid today as 25 years ago
- Problems are "pushed" downstream to after-sale services with smart products. Leads to increased work there, as the smart products are not smart yet!

The premise of this thesis is to facilitate the making of smart products by supplying a framework for structuring relevant knowledge.

### 3.1.3 ORGANIZATION VIEW

Organization in modern society has become the dominant way of "getting things done" when tasks are too large for a single person to achieve. This has had huge impact on us and our way of being, so analysing, understanding and making organizations better has been an object of analysis for about almost a century. There are three approaches to organizations and their analysis: Rational, Natural or Open-system approach. Each has its merits and drawbacks. They all deal with grouping people to accomplish some task, and they agree that people, purpose / goals, structure, hierarchy, power etc. are part of the mix. What they do not all agree on is how the mix is to be, and what is "clear" and what is not.

These approaches came into being in chronological order: first rational, then natural and open; they start with simple, and move to more complex and all encompassing viewpoints. Just like a human being starts by securing food before thinking about ego or self-motivations. Just like physics, from Newtonian to Relativity to Strings to the theory of everything. It would not make much sense to go the other way around. It is about sense-making, our limited cognitive abilities, our attempt to understand our surroundings.

Three main archetypes for looking at organizations are here taken from (Scott 2003):

Rational system:

> Organizations are made with a purpose and a goal that have been agreed upon by the participants. Participants are organized into relatively highly formalized structures.

Natural systems:

> People have different and more complex agendas with their participation in organizations. Goals and purpose are considered more *diffuse* than in the rational approach and one overriding goal is always present: survival of the organization. Structure is not as formalized and informal ties between participants guide behaviour more than formalization.

Open system:

> Organizations are not isolated from the environment but interact with it and depend on it for their own survival. Open systems are capable of self-maintenance on basis of throughput of resources from the environment. Two properties of open systems are morphostasis and morphogenesis, or the ability to maintain a given form/state/structure and the ability to elaborate or change the system. Participants in open systems are less coupled and can be part of a coalition, and these coalitions are in constant reconstruction, both with regard to goal and participation.

These three ways to understand an organization are still open to discussion. What is important here is that they exist and that they depend on where the "system boundary" is drawn. Most efforts in understanding organizations and especially in engineering rely on both humans and organization to be rational. Let us look a little closer at rationality and three authors' views on it.

Simon (Simon 1945) on "Rationality":

> Focus on means-end thinking; decisions can be viewed as either *value premises* based on what ends are preferred or desirable, or as *factual premises* based on how to get there. There is a relative element in decisions that implies, because of human limitations, that persons can never consider all options or all effects of decisions; they must focus on organizational things and choose from a subset of possibilities. This often leads to sub- optimized decisions that are "good-enough". Finally, efficiency is imprinted in all employees from the start.

Weber (Weber 1924) on "Rationality":

> Two-sided view of rationality: the structure of the cooperation and the management of employees. The offices are organized in a hierarchy, where each has a specified and clearly defined sphere of competence. Obedience of employees is not to persons but to impersonal order / role / function. Written records of acts, decisions and rules are kept. The second aspect is the management of employees. They are free persons, selected on basis of technical qualifications, appointed, not elected; and there is complete absence of appropriation for office. Employees enter free contractual relationships that they can leave at any time, but they can be promoted by seniority and/or achievement. They are then paid fixed money salaries.

Taylor (Taylor 1911) on "Rationality":

> Taylor states that by meticulously gathering all relevant knowledge of the task from all actors and formulating that knowledge explicitly, it is possible to identify ways to make the task a lot more efficient. Similar scientific methods are to be used to find the best workers, then further develop them and match them to the newly designed tasks. The by-product of this "optimization" is increased management.

As so many times before, there is no "black" and "white" but only different shades of "grey". It is clear that there are no "clean" systems, and even though the archetypes look very different, they are somewhat connected. Several suggestions have been made about the connection between the aspects; some say different organizations are different types, depending on what they do and in what industry they find themselves.

Others say that all system types manifest themselves within the same organization, depending on the department: rational for lowest level or technical and operation levels, natural for management level, and open systems approach for institutional level. Yet others state that rational and natural views are extremes on the same scale. Rational versus natural views are like capitalism versus socialism, very few exist at the extremes; most are only different mixes.

Why has the focus been changing from closed to open, rational to natural? Maybe it is part of human nature. We (as people) like to be in control; we like to control, and we conceive of things better as simplified rational things. Even though we rarely are in control, we like to believe we are, and hence we try to make our environment (here organizations) more *simple* and *rational* so we can feel in control. This also aids in interpersonal communications – communicating a rational system is much easier than a natural system when introducing new personnel to the system. We realize that the phenomenon of the organization cannot be described with rational thinking alone, but it is only when we have *mastered* the old approach that we move to the next. This is again like Maslow's pyramid of needs (Maslow 1943), one only moves up the levels as they become satisfied. Another aspect is the changing world: as we have more than enough of everything, we start to focus on *softer* things. Operational efficiency gives way to human values and focus in research.

In the researchers' mind, one of the best works of combining the different aspects and matching them to the actual world is the work of James Thompson in his book *Organizations in Action* (Thompson 1967). Thompson's main point is that organizations will strive to be rational even though they are both natural and open systems, depending on where we look. They will seal off the core technologies from environmental influences and try to control input and output transactions (buffer, smooth, forecast or ration, in that order).

The organization will try to gain power over relevant elements of the task-environment, and it will do so by seeking alternatives and prestige, making contracts, co-opting or coalescing. If all fails, it will seek to enlarge its task-environment. Part of the power play is to place their boundaries for the task-environment in such a way that the would-be crucial contingencies are incorporated; how will depend on the technologies the organization uses. The organization will try to balance its components (departments / subunits) by using the size of the one that is hardest to reduce to size the others. It will also try to use its excess capacity by enlarging their domains.

Three types of component interdependence are found in organizations: generalized, sequential and reciprocal. Pooled (generalized) interdependence is where a component does not depend directly on another but is linked through a pool. If one performs badly, it will affect the others in the end (on the bottom line), and generalized interdependence solved by standardization (the least costly way). Sequential interdependence, where one component is dependent on the next, is solved by planning (mediate cost); and reciprocal interdependence, where output from both is input to both, is solved by mutual adjustments (most expensive). Cost is the measure of communication and decision effort. Organizations try to minimize cost by localizing and making conditionally autonomous, first reciprocal, then sequential, and finally grouping into homogeneous groups to facilitate standardization.

Structure of the organization will depend largely on the task-environment and how it is put together. There are two dimensions: the variety of the environment (from homogenous to heterogeneous), and how fast it is changing (from stable to dynamic/shifting). To deal with the first dimension, organizations will try to make the environment homogenous with its structural units, which can lead to homogenous segmenting of a heterogeneous environment and an equal number of units to deal with it. The second dimension tells how the organization should deal with changes in the environment. In a stable environment, rules will suffice, in a known variation a standardizing sets of rules, but when the environment is too large or unpredictable, localized monitoring of the environment is necessary and a plan made accordingly.

The closeness of the technical-core and boundary-spanning activities will greatly influence the organizational structure. When they

> *...can be isolated from one another except for scheduling, organizations will be centralized with an overarching layer composed of functional divisions.*
>
> *(p.75) in (Thompson 1967)*

If too many major components of the organization are reciprocally interdependent, segmentation and self-sufficient clusters within their own domain will be made. Thompson then finishes by dealing with specialized organizations (like consultants):

> *Organizations designed to handle unique or custom tasks, base specialists in a homogeneous group for "housekeeping" purposes, but deploy them into task forces for operational purposes.*
>
> *(p. 80) in (Thompson 1967)*

Where does all this talk of organizations and different views take us? It has been a process to narrow down the field of organizational research to some major points that relate to this thesis. The issues that serve as rationale as to why this thesis proposition should be put forward can now be summarized. A recap of organizational impact on the problem is as follows:

- Rationalism still dominates in planning production and other activities that are "enclosed" in the organization.
- Efficiency is still the main mantra and a systems view is not used.
- Organizations try to hide their core from outside influences.
- Effects of open and natural systems are usually not included.
- People are knowledge bases but are "free" and often leave the organization, hence taking their knowledge along.
- Organizations "behave" like organisms with prime focus on self-preservation.

Organizations are comprised of people. These "atoms" add yet another dimension to be examined when trying to understand the impact of the problem.

### 3.1.4 PEOPLE VIEW

This view deals with two main issues that are of relevance here, people as machines with focus on strength and limitations, and the process of learning. Looking at the human factor, we immediately recognize that we are limited, but also that we are quite capable in circumventing our limitations without noticing.

It has been shown that humans have a limited attention span, and one of its best descriptions is *the magical number seven* (plus or minus two) as discovered by (Miller 1994) in 1956. He states that short-term memory is only capable of handling seven "chunks" simultaneously, plus or minus two, depending on the person. The chunks can be different – if they are "coherent", they can be whole words or concepts, or even larger things. If on the other hand they are not coherent, chunks become smaller and smaller, ending in single letters, numbers and the like, again depending on how coherent they are. This limitation of our "processing capabilities" should be recognized and kept in mind when designing for human beings.

Another thing to remember is our brain and its "visual capacity". Looking at our brain (Blakemore & Frith 2005), we find that it has four lobes. The frontal lobe deals with logic (among other things), while the occipital lobe at the back is devoted to vision, and the parietal lobe above that deals with movement, position, orientation and calculation (Figure 28).

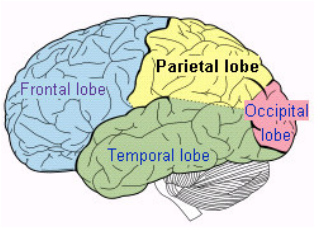



**Figure 28 - The brain and its lobes**

So, a really big portion of our brain deals with images, especially three-dimensional images, and the area of the brain that processes images is not the same as the one dealing with logic. If knowledge can be presented in a visual way, it may be easier to interpret. It is worth noticing that an image can be a chunk. And we all know that a picture tells as much as a thousand words. Explicitness of the relevant knowledge is also important. Humans rely a lot on *tacit* knowledge (Polanyi 1962) and understanding thereof. This thesis is not going to join the debate on tacit knowledge. We only point out that humans know more that they can formulate, and for this problem, as for most knowledge engineering problems, this should never be forgotten. How humans actually solve problems is relevant and has been studied in dept in psychology and variants thereof. Two very good studies are: *Human problem solving,* by (Newell & Simon 1972) with their heuristic versus "complete" method, and *Unified theory of cognition* by (Newell 1990). An aspect of problem-solving methods is how communication plays a part. An excellent study on human communication is the sense-making theory by (Dervin 1998) in which the importance of communicating "wide" is made clear. We return to sense-making theory later, when we use its arguments.

Learning and applying knowledge is a human trade. In this thesis, the plan is to make a system "more intelligent" so that much of the manual labour now being done will disappear. To identify and structure the knowledge needed, let us look at how humans learn. One way to "measure" or evaluate learning is the *Structure of Observed Learning Outcome* (SOLO) taxonomy (Biggs & Collis 1982). SOLO is a model that describes levels of increasing complexity in a student's understanding of subjects. It was proposed by John B. Biggs and Collis, (Biggs & Collis 1982) and has since gained popularity. The model consists of five levels of understanding:

- *Pre-structural* - The task is not attacked appropriately; the student has not really understood the point and uses too simple a way of going about it.
- *Uni-structural* - The student's response only focuses on one relevant aspect.
- *Multi-structural* - The student's response focuses on several relevant aspects, but they are treated independently and additively. Assessment at this level is primarily quantitative.
- *Relational* - The different aspects have become integrated into a coherent whole. This level is what an adequate understanding of some topic normally means.
- *Extended abstract* - The previous integrated whole may be conceptualized at a higher level of abstraction and generalized to a new topic or area.

Comparing this thesis and its premises to SOLO, it can be said to involve a move from multi-structural to relational. Instead of looking at individual aspects, we want to relate all aspects and create a whole. Humans play a crucial role in this, and it is therefore very important to remember that we are limited creatures. So if it is possible to use our facilities better, we should consider that. A recap of our role in this as humans is:

- Limited capabilities, short-term memory handling of 7 +/- 2
- Great visual capability of humans
- Adaptability beyond everything, unconscious
- Five learning levels that translate into how to present the problem
- Constant sense-making
- Tacit knowing, "having it in the hands", very important in after-sales

This concludes the four aspects of the problem. Let us now summarize the main points and formulate a rationale for why we shall solve this problem.

### 3.1.5 SOLUTION INPUTS

Business today focuses a lot more on after-sales service, and new buzz words include product-service systems (Alonso-Rasgado et al. 2004). A great focus on operational efficiency in production has pushed problems into after-sales, and organizations are now feeling the consequences. Organizations are dynamic "beings", comprising "free" people who come and go. Keeping knowledge within the organization is an increasing problem, a problem that has spawned a whole field of research, Knowledge Management. As most people who have been involved in changing organizations know, it is a hard, time-consuming process that is not always successful. To spice things up even further, organizations are made up of people with limited capabilities who constantly need to be making sense of complex situations.

To state our view explicitly, we want to suggest a solution that does not burden the organization. So any across-the-functional-boundary solution in the form of a monstrous IT system is not an option. We also like to utilize human capabilities better, or rather circumvent our limitations and draw on more of our strengths. With all this in mind, let us formulate a reason for dealing with the problem.

### 3.1.6 WHY SOLVE IT?

Having formulated the problem and its aspects, highlighting the whys is now in order. We will deal with the hows later in this chapter. We can say that there are two views of the reason the problem should be solved: the money aspect and the resource aspect. They are of course coupled, but let us state them as the success criteria of the case company. Success criteria for solving the problem:

- Installation complexity (as time and security)
    - Ease-of-use through eliminating repetitive labour
    - Security, i.e. only legal solutions available and inability to select illegal solutions
- Reduce installation failures
- Reduction of implementation cost

The case company has identified the problem and linked it to the lack of system view in product construct. Elements also recognized to be part of the suggested solution are:

- transfer to system view
- make system transparent, generate overview
- support Life-Cycle evolution
- new module support along with new function support

When a solution is suggested later in this chapter, these elements must be kept in mind. The problem actually coincides with work done in AI a decade ago. The core rationale for this thesis can be better described with an "immobot". Found in a great work done at MIT, *Computer Science and Artificial Intelligence Laboratory* (CSAIL), *immobots* were defined as:

> *...the information gathering capabilities of the Internet, corporate intranets, and smaller networked computational systems supply additional testbeds for autonomous agents of a very different sort. These testbeds, which we call immobile robots (or immobots), have the richness that comes from interacting with physical environments, yet promise the ready availability associated with the networked software environment of softbots.*
>
> *(p.1) in (Williams & Pandurang Nayak 1996)*

Williams and his co-workers share our rationale and are noteworthy because of their focus on implementation and not knowledge acquisition. Williams' words for the solution we need to achieve are very appropriate:

> *Finally, we argue that these immobots give rise to a new family of autonomous agent architectures, called model-based autonomous systems. Three properties of such systems are central. First, to achieve high performance, immobots will need to exploit a vast nervous system of sensors to model themselves and their environment on a grand scale. They will use these models to dramatically reconfigure themselves in order to survive decades of autonomous operations. Hence, self-modeling and self-configuration comprise an essential executive function of an immobot architecture. Second, to achieve these large scale modeling and configuration functions, an immobot architecture*

*will require a tight coupling between the higher level coordination function provided by symbolic reasoning, and the lower level autonomic processes of adaptive estimation and control. Third, to be economically viable immobots will have to be programmable purely from high-level compositional models, supporting a "plug and play" approach to software and hardware development.*
*(p.2) in (Williams & Pandurang Nayak 1996)*

The researchers could not formulate this better. So, how do we achieve this?

## 3.2 INSPIRATION SOURCES

After analysing the different aspects of the problem, some things are apparent here. We want the solution to be relational so that it screams *system theory* as it focuses on relations instead of the thing itself. We are also quite focused on products "knowing more" themselves. This points toward *artificial intelligence* and especially the practical and most successful incarnation of AI, *knowledge-based systems*. We recognize that the problem is of a software nature, so *software engineering* is an obvious place to look. All of this is also embedded in the actual design of the product, which is *engineering design*. A quick look at these sources is in order.

### 3.2.1 SYSTEM THEORY

System theory was born in biology and the observation of what nature is like. This is theoretical and has evolved parallel to *cybernetics* and *computer science*. Its most practical application can be found in *systems engineering*, which also serves as inspiration here. This thesis deals with systems, hence a deeper look at the different aspects of system thinking.

### 3.2.2 KNOWLEDGE-BASED SYSTEM (KBS)

Wanting the system to have more "knowledge" can be realized with a KBS. There are seven main types of such systems (expert system, neural network, case-based reasoning, genetic algorithms, intelligent agents, data mining and intelligent tutoring systems) (Kendal & Creen 2007). Of these, two are of main interest, *expert system* and hereunder configuration systems, and *intelligent agents*.

We examine these two types of KBS with most of our efforts spent on the process of making KBSs, the *knowledge engineering* (KE) process, and how it can contribute to a suitable solution. Most techniques used in KBSs stem from *Artificial Intelligence* (AI); lets call them "mature" techniques of AI. The difference between KBS and AI can be defined as follows:

*Artificial intelligence aims to endow computers with human abilities. ... Knowledge engineering, on the other hand, is the practical application of those aspects of artificial intelligence that are well understood to real commercial business problems...*
*(p.21) in (Kendal & Creen 2007)*

As there are no directly applicable tools in KE, we also have to look at the "mother" in order to gain inside information and hopefully inspiration.

### 3.2.3 ARTIFICIAL INTELLIGENCE (AI)

From the field of artificial intelligence, we wish to seek some inspiration. We aim to find building blocks within AI that are applicable to more practical application. General understanding of the AI field (Russell & Norvig 2003), thoughts on where it is going (Simon 1991;Simon 1995), rationale for why to use it (Simon 1996), and actual techniques (Hopgood 2000) comprise the search method here. It is worth stating though that much of AI work is on technical things, and as this thesis is about knowledge representation, we try to fish out similar aspects from AI. This is actually linked to cognition and the analysis of human capabilities, since that is what AI tries to emulate!

### 3.2.4 SOFTWARE ENGINEERING

A fine overview on *software engineering* (SE) is given by (Sommerville 2007). By analysing the field, we can state the following:

- Software Engineering is very technical-solution oriented and often does not seem holistic in its approach.
- Knowledge Representation techniques are visually impaired, too logic- driven, and Unified Modelling Language (UML) is not so connected to technical solutions.
- The researchers have a problem with object-oriented-design (OOD), find it confusing, fussy, not driven by "understandable criteria", and often way too abstract. This might be rooted in the fact that researchers have not seen many (read any) good OOD solutions, seen from a documentation perspective.

We are not saying that SE is without merit, just that its focus is quite different from what we want to achieve. There are some sub-fields we would like to look into further, as they seem to touch on relevant issues. These are the fields of *Requirement Engineering* (Hull et al. 2005), *Service-oriented Architecture* (Baglietto et al. 2005) and *Distributed Systems* (Coulouris et al. 2005). We are not dealing with SE as an inspiration source but rather these sub-fields.

### 3.2.5 ENGINEERING DESIGN

The design of artefacts is a core thing here. The core premise of this thesis is that a redesign is needed to implement changes. There is especially the sub-field of *functional modelling* that is of interest, as it tries to capture the purpose of artefacts and place them in context. Engineering design (ED) methods are also a source of inspiration. An important factor is the intersection of ED with both system thinking and software engineering.

## 3.3 THE SUGGESTED SOLUTION

The problem, different aspects of viewing the problem and understanding the problem as dealt with in the previous sections serve as catalyst to a solution suggestion. Using two current modelling techniques and joining them can be seen as a basis for a complete method for modelling a distributed embedded system. The suggestion is to join Product Variant Master (PVM) (Hvam et al. 2007) and Design Structure Matrix (DSM) (Steward 1981), and then add more aspects drawn from the inspiration sources. The PVM has visual strength, allows intuitive decomposition and is easy to learn (by analogy), while DSM is rigid and suited for visualizing relations. Together, the PVM and DSM should provide a comprehensive tool to model distributed embedded systems.

By building on system theory and its practical application in systems engineering a foundation is laid. General knowledge engineering is to be used for the main process, while bits and pieces are "borrowed" from AI, engineering design and software engineering. By standing on system theory, a focus on relation is established. Current work has been lacking in stringent relation definitions and handling, and this is one of the focus areas in this thesis. The knowledge modelling has to "connect" very well with software implementation so a rigorous notation is needed with the aim of allowing one-to-one mapping between model and programming. Another issue is the distributed character of the problem. The system has to be able to cope with new sub-systems and solve closed problems by itself. This implies that knowledge of some sort has to be "coded" into each sub-system, and a communication language has to be established.

The solution is therefore comprised of two main elements that form a concept. The concept is called *Embedded Configuration* with the elements of *encapsulating knowledge* and *communicating knowledge* between encapsulation units. Let us first look at the concept and then walk through the two elements.

### 3.3.1 EMBEDDED CONFIGURATION

Having KBS systems embedded into each sub-system is not new, not from a technical viewpoint at least. There are several commercial software systems available that are embedded rule-based systems (like *Array* (http://www.arraytechnology.com/, accessed 07.07.2008) and *ConfigIt* (http://www.configit.com/, accessed 07.07.2008). These suppliers have their own development environment, but it is focused on the implementation. A knowledge structure method could supplement these very well. In this thesis, we use the name *Embedded Configuration* to describe the concept of having several nearly independent sub-systems, each with a KBS system embedded, which have to form an complete whole. So, Embedded Configuration in this thesis has a wider meaning than might be deduced from the term alone. The definition of embedded configuration used here includes the following:

- Computer in each sub-system.
- A KBS in each sub-system, in the case company only rule-based systems but could probably be any kind.
- Knowledge encoded into each sub-system and not in data format but as information or knowledge.

- Mimic organs and hierarchically nested systems
- System is self-sufficient, meaning no external IT system. Has agent-like behaviour.
- Communication between sub-systems is kept to a bare minimum and what is communicated is well defined (with standards and on higher abstraction levels, not data transfers).
- Focus on meaning and knowledge, not technical implementation (hereunder protocols / basis language).

This allows us to define *embedded configuration* as follows:

**Definition IV - Embedded Configuration**

| Embedded configuration is a method to set up and maintain a system of agents, each of which has embedded KBS. The knowledge coded in the KBS is on the knowledge level, and communication to other agents is kept a bare minimum. The system is self-sufficient, does not rely on external computers, but it can receive inputs. |
|---|

With the concept in place, the requirements for the modelling technique are also set. It has to support the different aspects mentioned in the definition. The model suggestion follows in the next section.

### 3.3.2 MODELLING ENCAPSULATION OF KNOWLEDGE

Encapsulating knowledge into each sub-system is one of the main drives in this thesis. The hypothesis is that by increasing abstractions in the model, it is possible to seriously reduce the number of decision variables an outsider has to consider in order to set the sub-system to its needed stage.

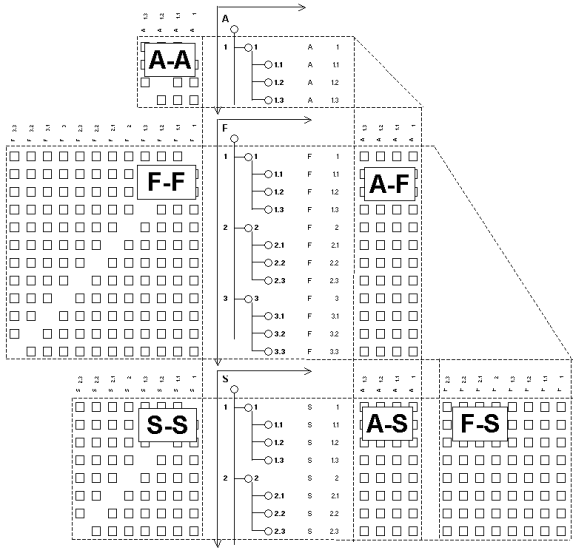



**Figure 29 - PVM with mapping and relation matrixes**

To do this, we suggest using the PVM with three abstraction levels: application (A), functional (F) and artefact (S) views, and stringently couple the levels together with DSM matrixes. There are both mapping matrixes (between levels, represented with different letters, e.g. A-F in Figure 29) and internal matrixes (within levels, represented with same letters e.g. A-A in Figure 29) to couple the views together. The PVM trees

are placed in the middle, with mapping on the right-hand side and relations on the left-hand side, as shown in Figure 29.

The basic thinking is that most aspects should be "internal", within one model and a "cascading" effect should be present in the model. By cascading, we mean that a single decision at top level should trigger choices down the levels through both mapping and relational matrixes. Items that need to "communicate" outside the model must be marked clearly. To aid in both communication and decomposition of the trees, ontology is probably very useful.

The model should generate a "palette", a picture, which should give a good visualization of the relational structure of each sub-system. It is hoped that an analysis method is to evolve through pattern recognition and hence build on human capability to deal with graphics. The model is also thought out to be "layered", depending on the viewpoint; here, details blur out when viewed from a distance and patterns appear. In close, details become more and more clearly visible when reading individual relations or branches of the tree. The last aspect here is the "holographic" attribute; this model can be drawn for a sub-system, the whole system or even for a system of systems. It looks the same; only names in branches and relations would change. From this attribute, a stacking could occur, meaning that we can draw a model for each sub-system and then merge them into a system model. As the model is the core of this thesis, it is described in detail in a later chapter.

The other main element of the concept of embedded configuration is communication between models. Let us ponder on communication of knowledge and how it could be modelled.

### 3.3.3 COMMUNICATING KNOWLEDGE

We constantly try to communicate knowledge to our surroundings. This might best be illustrated with an example:

> The other day I helped my mother move photographs from a digital camera to a PC. This should be a straightforward task. The problem (well, one of them) was that we live in different countries, so I guided her by telephone. This took a lot longer than expected, requiring many iterations and explanations. After we had finished the task, I came to think of how this actually related to my work and the communicating knowledge "dilemma". It made me ponder on the following: Let us assume that you are a computer-literate person with some IT knowledge, and you were to communicate your knowledge to an elder semi-computer-literate person who is to perform a simple (from your point of view) task. Would you prefer to do this by telephone in your own language, meaning that you could only use words and no observations or manual guidance, or on-site without knowing the other person's language and having to rely on mime, observation and manual guidance?

An interpretation of this scenario could be that in the first case, you would try to decode your knowledge into information and then ask the other person to carry out a task or an action that he or she does not understand as he or she does not have the knowledge to either interpret or put into context. It would be a lot like playing blind-chess and having a semi-independent "machine" to move your pieces, because people do not

always do precisely what they are told or the instructions are not precise enough. You would then have to check constantly whether the person had done what you asked him or her to do.

In the other case, you would not have to decode your knowledge or explain both interpretations and context. A simple "follow my lead" would suffice, even though you could not speak a word of the other person's language.

Every human undertaking is based on our knowledge of things and surroundings. We are often not very aware of our knowledge and how it is structured; we just use it. If we are to construct an intelligent system, we have to know what knowledge is, and especially how to transfer or communicate it to the machines. Communicating knowledge can draw inspiration from many different fields. Human beings do it all the time, and much research on this topic has been carried out. Computer science has tried to emulate humans in artificial intelligence, especially distributed AI and the design of complex control systems has focused on communication for quite some time. When we communicate in our daily lives, we usually do not differentiate between data and information on the producer side, or information and knowledge on the consumer side. Matthias Rauterberg (Rauterberg 2001) shows this well in his producer to consumer view of communications, as shown in Figure 30.

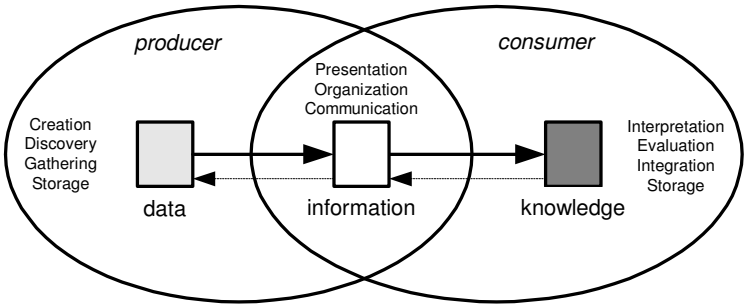



**Figure 30 – Producer / consumer communication (Rauterberg 2001)**

So, what gets communicated is information or context-rich data, and it is then subsequently the consumer who interprets the communiqué to knowledge. This producer / consumer terminology could be more easily understood by using communication theory which defines producer as sender and consumers as receivers (Figure 31).

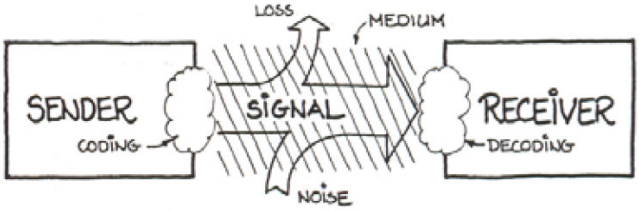



**Figure 31 - Communication theory (Shannon & Weaver 1949), taken from (Buur & Andreasen 1989)**

This is one of two other ways to look at communication, the other being communication layers (Figure 32). Even though these look different, they have a similar rationale. The lowest layer in Figure 32, the physical carrier, is the same as the middle part of Figure 31. The protocol decides how coding and decoding is done and subsequently what the signal is like. The two higher layers, Meaning and Flow of control, are not explicitly drawn in Figure 31, but the former, Meaning, would be

implicitly what the sender wants to say to the receiver. Flow of control is not present in the communication theory, but we could think of ways to add this to the picture.

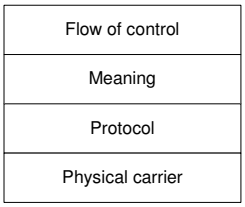



**Figure 32 - The communication layers (Lind 1990)**

In anthropology we find yet another way to look at communication, the high and low context communication put forth by (Hofstede 2001). This focuses on the awareness stage (as defined in (Ahmed et al. 1999)), and on how much knowledge / context has to be added to the actual "signal" to get the "right" meaning from the communication. Think about the following:

> Ask anyone in most western cultures for directions to a place that does not exist, and everybody (well, most likely everybody, apart from some jokers) will tell you that the place is nowhere to be found. Do the same in some middle-east countries, where it is considered rude not to help, and some people might try to guide you to somewhere.

The context of communication has another impact; it assigns a role to the sense-making process. Sense-making in communication has been researched, for example, in sense-making theory (Dervin 1998). Its core is "asking the right questions" when communicating. Think about the following:

> You go to the library and ask for Shannon's & Weaver's article on communication. The librarian gives you what you ask for, but you later realize that it was not exactly what you where looking for. If, in the beginning, you could explain to the librarian why you were there, like: "I am working on how to communicate knowledge and need literature related to communication", then the librarian would have had a chance to "interpret" or put into "context" what you wanted and find something relevant.

In relation to what has been said earlier, the former is communicating data and the latter is communicating information. When communicating information, the receiver has the possibility to "interpret" or "contextualize" it to multiple sets of data. This could be drawn onto the data, information and knowledge mapping figures with an arrow running down from knowledge to data. The lower the context barrier, the "easier" the mapping between data and information / knowledge ought to be. Like this paragraph on communication hints, knowledge would most likely not be in the communiqué, only information, and it would then be up to the receiver to make the interpretation.

### 3.3.4 MODELLING COMMUNICATION

Following the rationale presented in the earlier section, we can state that modelling communication has to rely on some points:

- Focus on meaning (not protocol or carrier)
- Use verb-noun combinations and ontology to dictate rules of engagement
- Use standards for naming, messages, protocol and especially error handling

- Mimic sense-making communication of humans, include higher abstractions and allow each sub-system to interpret.

The communication can only be tested with thought experiments until an artefact prototype is constructed.

## 3.4 HOW IS IT DONE?

What is suggested here is not a complete method for solving knowledge engineering projects. It is rooted in the general thoughts on the modelling process in knowledge engineering (Studer et al. 1998) and should be seen as a sub-method for the complete process. Methods like *Knowledge Acquisition Design System* (commonKADS) by (Schreiber et al. 2000) and methods for doing product configuration by (Hvam 2001), a subset of knowledge engineering, should be used as the overall process guideline. The method suggested in this thesis will supplement the *knowledge*, *communication* and *design* models in commonKADS and replace the *product analysis* phase in the other.

This thesis does not suggest an overall process for doing knowledge engineering but relies on others for that, like those mentioned above. Let us now move on to the theory foundation of this thesis, based on the inspiration sources identified in this chapter.

# *Chapter 4*

## THEORIES

This chapter is about the underlying theories. As one of the main goals of this thesis is to enhance the domain of knowledge engineering with concepts and ideas from several other domains, this chapter draws on content from these different domains. At first glance, the content presented here may look stretched and incoherent, but it will hopefully all come together in later chapters.

Structure of this chapter is matrix-like. There are three topics: *application*, *function* and *artefact*; then there are four aspects within each topic: *Decomposition*, *Relation*, *Communication* and *Modelling Technique*. To describe them and their interaction, we make three "walk-throughs" of these seven issues: first, through all seven in the form of knowledge engineering and engineering design; then, sequentially through each topic; and finally, through each aspect. Hopefully, this will aid in providing an overview of the theoretical background used. Table 6 shows the matrix structure used here.

**Table 6 - Structure of the theory chapter**

| Aspect / Level | Decomposition | Relation | Communications | Model technique |
|---|---|---|---|---|
| Application | | | | |
| Function | | | | |
| Artefact | | | | |

It is worth mentioning that this chapter is not a recitation of relevant theory but an interpretation of it in relation to its use in this thesis. In each section, we try to refer to relevant, easily accessible introductory material for those not familiar with the section's content. To show what each author deals with in his/her work, the same legend of symbols is used throughout the chapter.

**Table 7 - Comparison legend for content**

| Four degrees of content: | Icon |
|---|---|
| Not mentioned ("empty") | |
| Mentions it | ◔ |
| Deals with it in more detail | ◐ |
| Deals with it and suggests ways to apply | ● |

Seeking inspiration from several areas and domains has helped populate these issues. To give an overview of all the sources, a theoretical foundation model is presented in Figure 33.

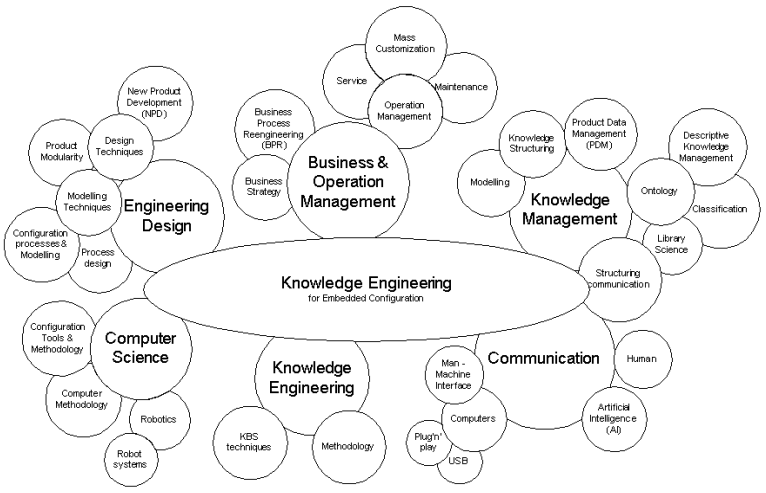
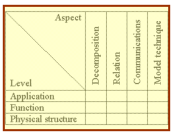



**Figure 33 - Theoretical foundation model (TFM)**

The items presented in the TFM are not handled separately but serve as a source for finding ideas and literature. It should be noted here that the subject presented in this thesis does not exist as a domain (or sub-domain) in current literature, at least not to the author's knowledge. Hence, this theoretical presentation serves to provide "building blocks" that can shorten the journey towards achieving *Knowledge Engineering for embedded configuration*.

The following presents general reviews of domains that deal with all seven issues mentioned in Table 6. There are three main fields identified as useful: first, system thinking; then, knowledge engineering; and finally, engineering design. They are presented here in the same order.

## 4.1 SYSTEM THINKING




In most science today, the issues are so complex that it is necessary to decompose the problems to smaller bits to be able to deal with them. This is especially the case when artificial things or artefacts are being constructed. When combining these artefacts into systems or complex solutions, another problem arises:

> *The whole is more than the sum of its parts (presumably first stated by Aristotle)*
>
> *(p.37) in (Klir 2001)*

therefore, identifying and dealing with the whole becomes tricky. The holistic (Smut 1926) approach is supposed to rectify this. This *top-down* approach with focus on relations between elements, rather than the elements (artefacts) themselves, makes system thinking quite interesting. The general notion here is that systems exhibit some fundamental attributes no matter what they are. Those attributes are the core focus of *system theory*. And therefore, by definition, system theory is very general and hard to apply. The discipline of systems engineering addresses the application of system theory. This section presents some important concepts from system theory and the relevant system engineering needed to apply system thinking. It is worth mentioning that development in system thinking is tied closely to evolution in automation (cybernetics) and computers. Now, let us look at system thinking.

### 4.1.1 WHAT IS A SYSTEM?

A system, as defined by (Bertalanffy 1969), is a set of elements with relations. A later definition by (Ackoff 1971) is:

> *A system is a set of interrelated elements. Thus a system is an entity which is composed of at least two elements and a relation that holds between each of its elements and at least one other element in the set. Each of a system's elements is connected to every other element, directly or indirectly. Furthermore, no subset of elements is unrelated to any other subset.*
>
> *(p.662) in (Ackoff 1971)*

What is considered an element or relation is totally reliant on the human perception and can be defined in an endless number of ways. A system is therefore based on our perception and does not exist on its own.

> *Every system is a construction based upon some world of experiences, and these, in turn, are expressed in terms of purposeful distinctions made either in the real world or in the world of ideas.* *(p.13) in (Klir 2001)*

This is a constructivist worldview. To aid us humans in understanding systems, several classifications have been suggested. Three classifications of systems that are likely to help this work are: the system level definition of (Boulding 1956), Jordan's system taxonomy (*Themes in Speculative Psychology*, Nehemiah Jordan, 1968) in (Skyttner 2001), and Klir's epistemological systems hierarchy (Klir 2001).

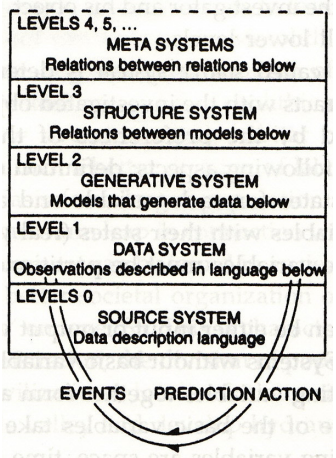



**Figure 34 - Klir's epistemological systems hierarchy, from (Skyttner 2001)**

These are important when identifying systems and categorizing them. Boulding suggests nine levels, ranging from *framework* to *transcendental*. In his hierarchy, levels two and three (*clockwork, cybernetics*) are the most interesting, as they cover most artefacts made by man. Jordan wants to use three organizing principles (*rate of change*, *purpose* and *connectivity*) with two poles each (*structural/ functional, purposive/non-purposive* and *mechanistic/organismic*), and his view is helpful because it focuses on purpose and connection. The most relevant way of classifying systems in relation to the work done in this thesis is the epistemological systems hierarchy by Klir. It introduces five levels (0 to 4) as shown in Figure 34. The focus of system theory differs from most disciplines as it is not on the elements themselves but the relations between them. System can be defined as:

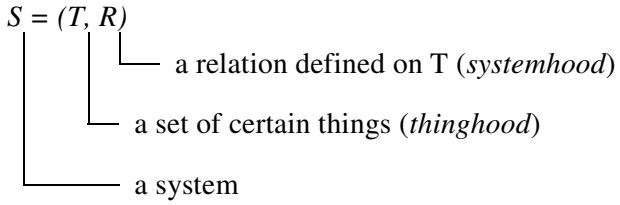



**Figure 35 - The *Systemhood* and *Thinghood* views of systems (Klir 2001)**

Thinghood is something most sciences deal with, but many sciences do not deal with the relation between them. This relationship focus is quite necessary in order to deal with complex systems. A short review of the most relevant concepts from system thinking now follows.

### 4.1.2 LITERATURE ON SYSTEMS

System thinking is a large domain with different aspects to it. For the purpose of this thesis, several concepts have to be introduced, since they influence the thinking process and affect the suggested solution. These concepts come from both the theoretical part of system thinking as well as the practical approaches available.

A very important aspect of systems is the concept of *equifinality* (Bertalanffy 1969), later named *purposeful system* by (Ackoff 1971) – i.e. systems that can reach a goal in many different ways, meaning that they are not mechanistic in nature as (Boulding 1956) defines his *clockwork* level, but adhere at least to his *cybernetic* level.

The concept of *feedback,* as introduced by (Wiener 1948) to use in control, and later made more general for systems by (Ashby 1956), is required for systems to be at least cybernetic in their behaviour. Feedback is the fundamental concept of our automated world today. It may seem trivial in the 21st century, but it has allowed great strides to be made in technical evolution.

Complexity is one of those concepts that all understand intuitively but few can explain. To aid in the quantification of system complexity, let us use Ashby's definition of complexity, or more precisely his notion of measurement of complexity, as:

> *the quantity of information required to describe the vital system.*
> *(p.2) in (Ashby 1973)*

Another aspect of complexity is *organized complexity* (Weaver 1948), where we have

> *...problems which involve dealing with a sizable number of factors which are interrelated into an organic whole.*
> *(p.536) in (Weaver 1948)*

Organized complexity allows for a way to deal with problems, since the randomness is not complete. One major aspect of this is beautifully stated by (Simon 1996):

> *Most of the complex structures found in the world are enormously redundant, and we can use this redundancy to simplify their description. But to use it, to achieve the simplification, we must find the right representation.*
> *(p.215) in (Simon 1996)*

This redundancy shows itself as a hierarchical nested system or systems within systems, or this is at least one of its appearances. This hierarchical structure helps us to decompose the system.

> *In hierarchic systems we can distinguish between the interactions* among *the subsystems, on the one hand, and the interactions* within *subsystems - that is, among the parts of these subsystems.* *(p.197) in (Simon 1996)*

Nearly decomposable systems is what hierarchically nested systems become when they are broken down, since the relations within each sub-system are much tighter than relations between sub-systems. This is noticeable all around.

> *One of the important properties that we observe in virtually all complex systems, whether they be social, technical or natural, is that they are nearly decomposable.* *(p.611) in (Simon 2002)*

Different approaches are chosen to solve a task depending on how a system is defined, with or without the human component, and with or without a clear goal. A hard systems methodology is when no (or very few) "fuzzy" elements (like human beings) are present in the system, and the system has a clearly defined goal. This is very typical for conventional engineering problems. The soft system methodology, for which (Checkland 1984) is one of the main advocates, is meant to deal with tasks where hard systems methodology fails. Checkland states that most complex problems and therefore complex systems are very seldom well defined with clear goals, and are therefore very hard to solve with hard methods.

A more narrow view on system is the one offered by (Hubka & Eder 1987) in the theory of technical systems (Figure 36).

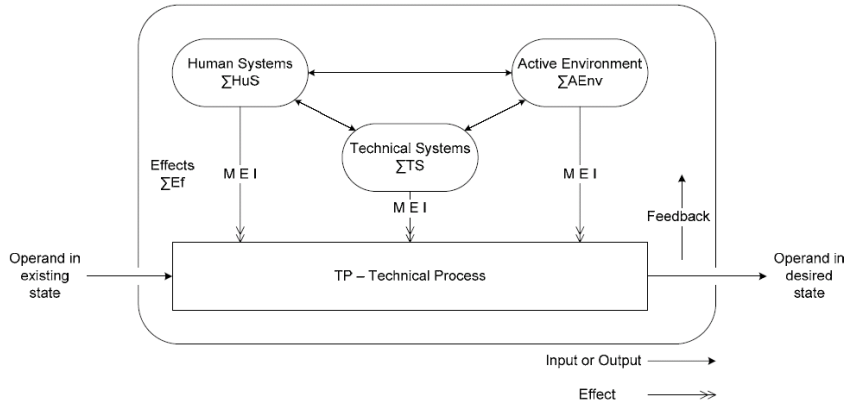



**Figure 36 - Theory of technical systems – redrawn from (Hubka & Eder 1996)**

Two interesting aspects of Hubka's theory are the inclusion of the human system and the breakdown suggestions for the technical systems with functions, organs and components. The former is largely left out of Hubka's treatment, as it would make the system "soft" (see Checkland); he therefore focuses on the technical system. The latter is a core concept that permeates this thesis throughout and is vital to all its rationale.

To be able to create or analyse systems, we have to realize what to look at and how to go about doing so. Now, we move to the practical aspect of system thinking. One of the early suggestions for the systems approach includes the five basics as introduced by (Churchman 1968):

1) *the total system objectives and, more specifically, the performance measures of the whole system;*
2) *the system's environment: the fixed constraints;*
3) *the resources of the system;*
4) *the components of the system, their activities, goals and measure of performance;*
5) *the management of the system.*

*(p.29) in (Churchman 1968)*

These thoughts still hold and have served as inputs to the domain of systems engineering. Systems engineering, the application part of system thinking (theory), defines it self as a three-part view of problems: *structure*, *functions* and *purpose*, as shown in Figure 37.

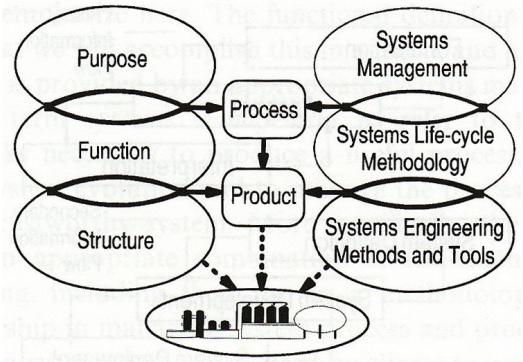



**Figure 37 - Systems engineering (Sage & Armstrong Jr 2000)**

Systems are defined in seven life-cycle steps. and within each of them. seven logical steps aid in defining each life cycle. The grand design is interpreted and evaluated in the V process model (with the same seven life-cycle steps), as shown in Figure 38.

The project management view on systems engineering (Stevens et al. 1998) is very helpful in deciding what to do and how. This work and its authors contributed greatly to the creation of ISO standards for systems engineering (ISO 2002) and its application (ISO 2003).

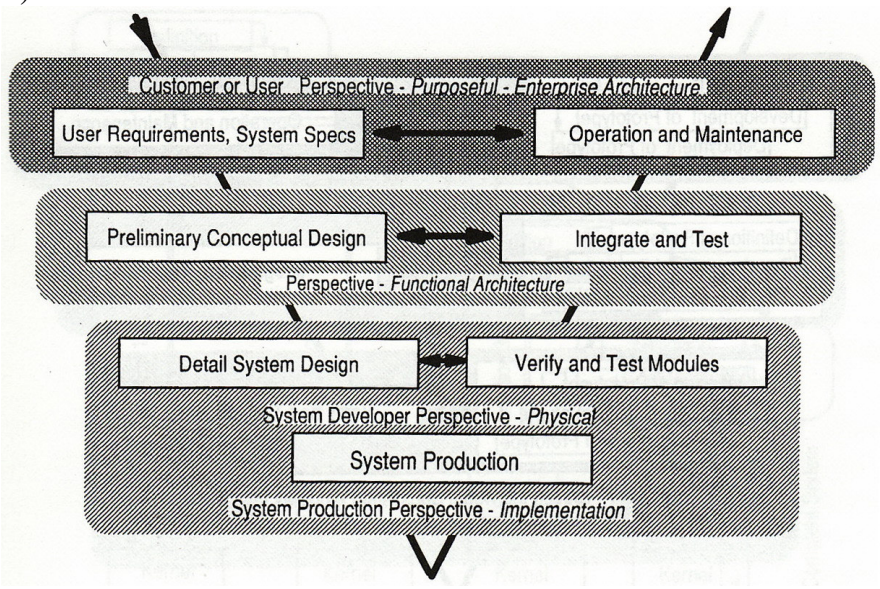



**Figure 38 - Interpretation of the grand design or V-process model (Sage & Armstrong Jr 2000)**

Business process literature is in many ways not that different in its approach. It tries to achieve a "holistic" way of looking at the firm and avoid getting stuck in the "conventional silos". An excellent view of different methods for doing BPR is presented by (Kettinger et al. 1997). A summary of selected literature regarding the aspects and levels presented earlier now follows in Table 8.

**Table 8 - Selected literature on system theory, system engineering and cybernetics**

| Levels & Aspects / Author | Application | Function | Artefact | Decomposition | Relation | Communications | Model technique |
|---|---|---|---|---|---|---|---|
| (Smut 1926) | ◔ | ◐ | | | ◐ | ◐ | |
| (Wiener 1948) | | | | ◔ | ◐ | ◐ | ◐ |
| (Boulding 1956) | ◔ | ◔ | ◐ | ◔ | ◐ | | ◔ |
| (Ashby 1956) | | ◐ | | | ◐ | ◐ | |
| (Churchman 1968) | | ◔ | ◔ | ◐ | ◐ | ◔ | |
| (Simon 1996) originally from 1969 | ◔ | ◐ | ◐ | ● | ◐ | ◐ | |
| (Bertalanffy 1969), (Bertalanffy 1972) | | ◔ | ◐ | ◐ | ◐ | | ◔ |
| (Ackoff 1971), (Ackoff 1973) | | | | ◔ | ◐ | ◔ | |
| (Checkland 1984) | ◔ | ◔ | ◔ | ◐ | ◐ | ◐ | ◔ |
| (Klir 2001) | ◐ | ◐ | ◐ | ◐ | ● | ◔ | ● |
| (Stevens, Brook, Jackson, & Arnold 1998) | ● | ● | | ● | ● | | ◐ |
| (Sage & Armstrong Jr 2000) | ◐ | ● | ● | ● | ◐ | ◔ | ● |
| (Skyttner 2001) | ◐ | ◐ | | ◔ | ◐ | ◐ | |

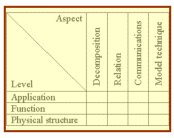
Some thoughts on the content/result in Table 8: As the reader can see, most authors do not deal with the selected topics in depth, meaning that they are trying to create a mind-set that is a bit different than that of other disciplines rather than making a "step-by-step" guide to making systems. This worldview is the systemhood or focus on relations rather than objects/things. The field of system thinking is still young and rather soft in its methodology. Most authors recognize this and teach us to be aware. Churchman states this much better then we can:

> *The essence of the systems approach, therefore, is confusion as well as enlightenment. The two are inseparable aspects of human living. (p.231) in (Churchman 1968)*

A general overview, which is a touch historical, of systems thinking is *General systems theory* by (Bertalanffy 1969) and *Facets of Systems Science* edited by (Klir 2001). A single book introduction to system theory is (Skyttner 2001), with a fine summary of different system theories (retold by the author), their use and methodologies to support them. The tool book to systems engineering is (Sage & Armstrong Jr 2000), while (Stevens, Brook, Jackson, & Arnold 1998) is the project approach.

A field where systems thinking and engineering is very relevant and could even be considered a sub-field is knowledge engineering.

## 4.2 KNOWLEDGE ENGINEERING




Manipulation of knowledge to be used in engineering tasks is not new. The world has been doing this for a long time. But it was first with the rise of computers and our attempts to make "intelligent" software systems like *Knowledge-Based Systems* (KBS) that the field of knowledge engineering was born. It is meant to help us formalize the process of constructing complex software systems that rely on codifying human knowledge in order to solve problems previously solved only by humans.

> *So the goal of the new discipline Knowledge Engineering (KE) is similar to that of Software Engineering: turning the process of constructing KBSs from an art into an engineering discipline. (p.162) in (Studer, Benjamins, & Fensel 1998)*

As the reader can see, the field of KE is quite broad and requires cross- discipline knowledge in order to be used successfully. KE, as explained earlier, is the focus on knowledge manipulation in relation to information system development, whereas such manipulation in relation to the organization is called *Knowledge Management* (KM). The two have strong ties and because this thesis deals with KBS and its creation, all knowledge manipulation is termed KE, even though some aspects are clearly KM or from other fields that study knowledge.

This chapter is about KE in its broadest definition, and it draws from the conventional knowledge engineering field along with knowledge management, knowledge acquisition, knowledge representation and fields that study knowledge as an entity in itself. It should represent "state-of-the-art" in manipulating knowledge for use in technical systems design where information systems are involved. Knowledge is a fuzzy word and no precise definition is out there. For the purpose of this thesis, it is necessary to look at the term and what it contains and create a consistent way to deal with it in the coming chapters. After doing this, we move on to look at selected literature on KE and how it contributes to this thesis. But first, what is knowledge?

### 4.2.1 WHAT IS KNOWLEDGE?

Knowledge has been a research topic for some time now. What is interesting is that authors focus on different aspects in their understanding of knowledge and its dependency on data and information. Note that these views are not contradictory; they just represent different viewpoints and seem to be dependant on the domain from which the author focuses his or her research. A quick and by no means complete look at the dependency between data, information and knowledge is shown in Table 9.

**Table 9 – Different authors on data, information and knowledge**

| Author | Data | Information | Knowledge |
|---|---|---|---|
| Moore | Digital object<br>Objects are streams of bits | Any tagged data, which is treated as an attribute<br>Attributes may be tagged data within the digital object, or tagged data associated with the digital object | Relationships between attributes<br>Relationships can be procedural/ temporal, structural/spatial, logical/semantic, functional |
| Wiig | | Facts organized to describe a situation or condition | Truths and beliefs, perspectives and concepts, judgements and expectations, methodologies and |

| | | | |
|---|---|---|---|
| | | | know-how |
| Nonaka and Takeuchi | | A flow of meaningful messages | Commitments and beliefs created from these messages |
| Spek and Spijkervet | Not yet interpreted symbols | Data with meaning | The ability to assign meaning |
| Davenport | Simple observations | Data with relevance and purpose | Valuable information from the human mind |
| Davenport and Prusak | A set of discrete facts | A message meant to change the receiver's perception | Experiences, values, insights, and contextual information |
| Quigley and Debons | Text that does not answer questions to a particular problem | Text that answers the questions who, when, what, or where | Text that answers the questions why and how |
| Choo et al. | Facts and messages | Data vested with meaning | Justified, true beliefs |
| Jensen Group | Representation of facts | Data plus Meaning<br>Understanding of patterns, relationships | Information plus Beliefs, Commitments, Assumptions, Design for application |

Constructed from Stenmark (Stenmark 2002), R. Moore lectures at Rice University, Houston, Texas, and Jensen Group (The Jensen Group 1997), who compiled from Fahey, Nonaka, Wurman and Gange.

Some interesting aspects in Table 9 are noteworthy. Although most agree on data as facts or symbols and information as data with meaning, the authors reach different abstraction levels in their knowledge definitions. The IT view offered by Moore has knowledge with "lower" abstraction than e.g. Choo's "justified true beliefs". The views that would make most sense in this article are the ones presented by the authors Quigley and Debons fused with Nonaka and Takeuchi. The views represented in Table 9 are quite well summarized in Mueller & Schappert (Mueller & Schappert 1999) as a set of abstraction levels, with a general description of each level. Linkage is pointed out as the most important aspect, not the definition of each level.

**Table 10 - General view on data, information and knowledge (Mueller & Schappert 1999)**

| In the classical interpretation: | That is: |
|---|---|
| *Data* is associated with syntax | *Data* per se has no meaning and may be seen as raw material for information. |
| *Information* corresponds to semantics | *Information* is context sensitive and meaningful in the sense that it is interpreted data. |
| *Knowledge* takes the pragmatic part | Since context is user (application) dependent, information can then be enhanced by its use, i.e. pragmatic *knowledge* |

Thus, saying how to move between the levels is more relevant than giving a precise definition of the same levels. A definition of each level is shown in Table 10. Taking this view of the linking and picturing would result in Figure 39.

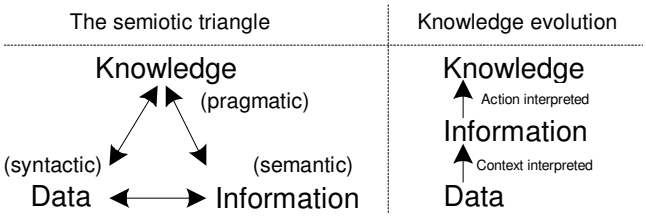



**Figure 39 - Relating data, information and knowledge (Mueller & Schappert 1999)**

So, where does this lead? Regardless of which view of data, information and knowledge is taken among those presented in Table 9, focusing on how to move between levels and identify what is essential should make it possible to make more intelligent systems. The syntactic and semantic are linked with context mapping, while

the semantic and pragmatic are linked to action interpretation, which can be even more useful if we use the terms context and interpretation, as pictured in Figure 40.

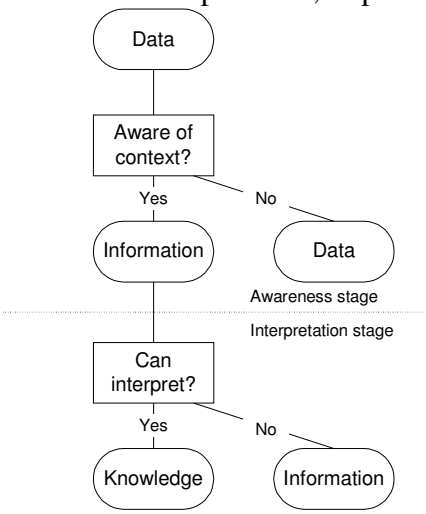



**Figure 40 - Moving between data, information and knowledge – redrawn from (Ahmed, Blessing, & Wallace 1999)**

The process of moving from data to knowledge (Ahmed, Blessing, & Wallace 1999), as indicated in Figure 40, should hold true for all viewpoints stated in Table 9. But, for the purpose of structuring data, information and knowledge in relation to embedded configuration, we focus on context and interpretation and use the definition in Table 10 as our guide. This actually coincides with the point of view offered by (Schreiber, Akkermans, Anjewierden, Hoog, Shadbolt, van de Velde, & Wielinga 2000), who state that maybe a precise definition of knowledge is not needed to be able to manage knowledge and its communication. On this premise, we move on to look at knowledge engineering literature. No single definition of knowledge is chosen here, but we focus on how to move between the trio of data, information and knowledge.

### 4.2.2 LITERATURE ON KNOWLEDGE ENGINEERING

This chapter deals with literature that can support the creation of an embedded configuration system from a higher abstraction level. It deals with at least two aspects. Literature on knowledge related to a specific topic or aspect is presented in the relevant chapter. The material presented here is drawn from Knowledge Engineering in its broadest definition. The core of knowledge engineering and software engineering is for that matter what Allen Newell calls the *Knowledge Level* (Newell 1982) in his Knowledge Level Hypothesis:

> *There exists a distinct computer systems level, lying immediately above the symbol level, which is characterized by knowledge as the medium and the principle of rationality as the law of behaviour.* *(p.99) in (Newell 1982)*

At the knowledge level, a principle of rationality is defined thus:

> *If an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action.* *(p.102) in (Newell 1982)*

This ties together the knowledge and goals of agents and their respective actions, without stating the mechanism through which the connection is made. To make this mapping explicit and complete, it is necessary to look at this connection in detail. So, it can be said that the core premise of this thesis is the relationship focus and mastery of its veiled being. The main facet of knowledge engineering is the incorporation of many views or aspects when dealing with construction of knowledge-based systems – in other words, its holistic approach. Most authors agree on the need for multi-abstractions in modelling systems, and the rationale for supporting this does not differ very much. Where the difference is most apparent is in how to connect these abstraction levels all the way to the intended application and guideline for decomposing the system.

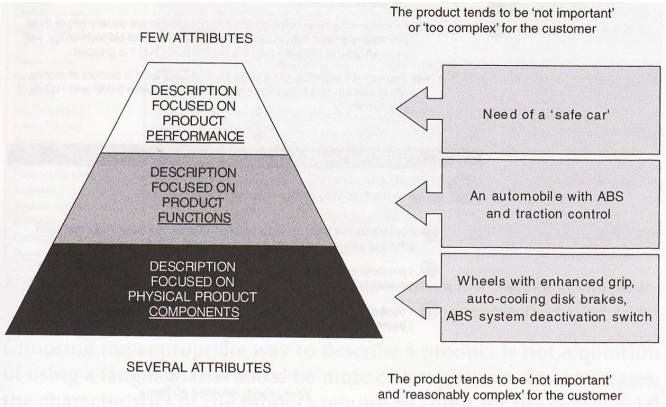



**Figure 41 - Product description with different degrees of abstraction (Forza & Salvador 2007)**

The different levels of abstraction in product description comprise an important facet of knowledge engineering. This is shown quite well in the illustration from (Forza & Salvador 2007) shown in Figure 41.

Knowledge representation and acquisition are important when trying to achieve a multi-level view of products. Forza & Salvador suggest two sets of models to solve this, the technical and commercial models (Forza & Salvador 2007):

> ***Commercial model**: a formal representation of the product space and of the procedures according to which a commercial configuration can be defined within such space.*
> *(p.53) in (Forza & Salvador 2007)*

and

> ***Technical model**: a formal representation of the links between commercial characteristics and the documents that describe each product variant (bills-of-material, production and assembly cycles etc.)*
> *(p.53) in (Forza & Salvador 2007)*

These models indicate where to acquire the relevant knowledge, but authors do not deal with these concepts in depth, and it is left up to the reader to figure out how to do this. The same holds true for the method suggested by (Hvam, Mortensen, & Riis 2007), where modelling is highlighted as very important but less munitions are used on how to decompose and relate elements.

The CommonKADS methodology by (Schreiber, Akkermans, Anjewierden, Hoog, Shadbolt, van de Velde, & Wielinga 2000) tackles all aspects needed to construct an embedded knowledge system, as shown in Figure 42 and Figure 43.

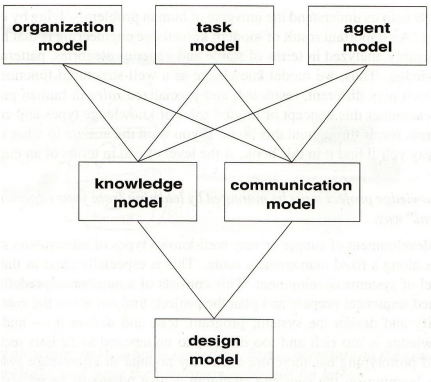



**Figure 42 - CommonKADS model suite**

The construction of knowledge models is described in three stages: *knowledge identification, knowledge specification and knowledge refinement* (Schreiber, Akkermans, Anjewierden, Hoog, Shadbolt, van de Velde, & Wielinga 2000), and it is mainly in relation to the last two stages that this thesis makes its contribution. A summary of major ontology approaches and their usage is presented in (Gomez-Perez et al. 2004); it gives us a very good look at ontology and its application.

What is helpful in their presentation is the view of communication and how to construct it in a systematic way. Communication between sub-systems in a system-of-systems is very relevant, and this thesis builds heavily on both the commonKADS and ontology approaches.

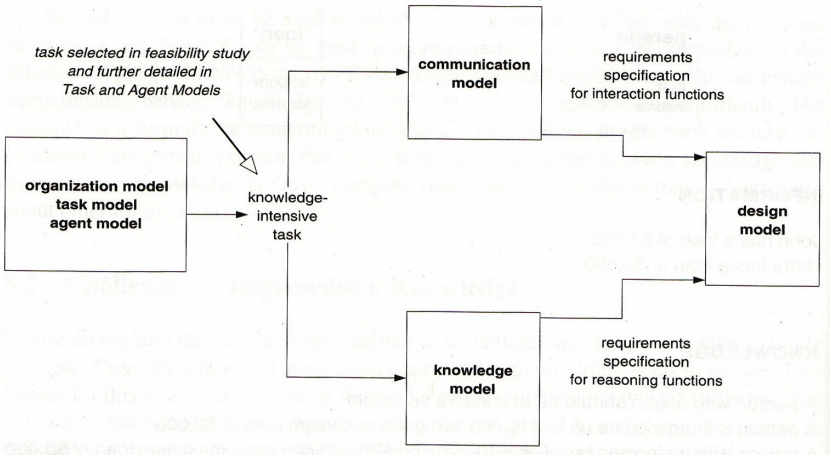



**Figure 43 - Schematic view of the role of the knowledge model in relation to the other models**

An important aspect of communication is introduced in commonKADS as the typology of transfer functions, and it is based on the *initiative* and *information-holder* dimensions, as shown in Figure 44:

| | | System Communication Initiative: External | |
|---|---|---|---|
| | Information Storage | | |
| External | | Obtain | Receive |
| Internal | | Present | Provide |

**Figure 44 - The *initiative* and *information-holder* dimensions, redrawn from (Schreiber, Akkermans, Anjewierden, Hoog, Shadbolt, van de Velde, & Wielinga 2000)**

Once these two dimensions are identified, we can predefine *communication types* and *conversation policies* to aid in the description of the communication needed. An example of types is shown in Figure 45.

| **Communication model** | **Predefined communication types** | | |
|---|---|---|---|
| **Task delegation** | | | |
| Request | Require | Order | Reject-td |
| **Task adoption** | | | |
| Propose | Offer | Agree | Reject-ta |
| **Pure information exchange** | | | |
| Ask | Reply | Report | Inform |

**Figure 45 - Predefined communication types**

An example of policies is shown in Figure 46.

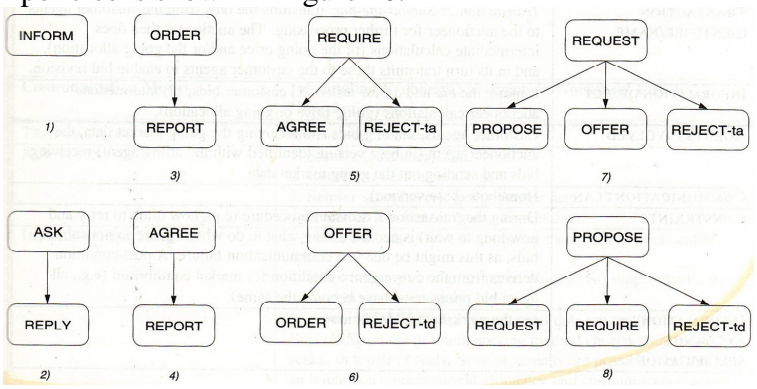



**Figure 46 - Communication type patterns**

It is assumed that these are quite general and can be used to name policies in all domains. Configuration definition also applies in this thesis, as defined by (Mittal & Frayman 1989):

> ***Given**: (A) a fixed, pre-defined set of components, where a component is described by a set of properties, ports for connecting it to other components, constraints at each port that describe the components that can be connected at that port, and other structural constraints; (B) some description of the desired configuration; and (C) possibly some criteria for making optimal selections.*
> ***Build**: One or more configurations that satisfy all the requirements, where a configuration is a set of components and a description of the connection between the components in the set, or detect inconsistencies in the requirements.*
> *(p.1396 ) in (Mittal & Frayman 1989)*

This is a central theme, as configuration should be seen as a key tool to implement the modelling techniques suggested in this thesis. Let us now look at selected literature on knowledge engineering and how it covers the three levels and four aspects introduced at the beginning of this chapter.

Most authors in Table 11 agree that multiple abstractions are preferred when describing the product. Very few of them actually offer any guidelines on how to actually achieve this, but talk generically about these levels. As a consequence, the authors do not take

decomposition and relation into account; some mention it in passing but most mostly ignore it. It is worth mentioning that even though some authors seem to address all seven issues, no single one can suffice to structure the content of this thesis. An amalgamation of a big portion of the concepts introduced in Table 11 would be ideal for further work.

**Table 11 - Selected literature on knowledge from an engineering perspective**

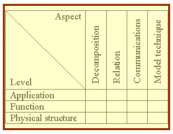
| Author \ Levels & Aspects | Application | Function | Artefact | Decomposition | Relation | Communications | Model technique |
|---|---|---|---|---|---|---|---|
| (Newell 1982) | ◐ | ◐ | ◔ | ◔ | ◐ | | |
| (Mittal & Frayman 1989) | | ◐ | ◐ | ◐ | ◐ | | |
| (Heinrich & Jungst 1991) | ◐ | ◐ | | | ◔ | | |
| (Nonaka 1991), (Nonaka 1994) | ◔ | | | ◔ | | | |
| (Akkermans et al. 1993) | | ◐ | | ◐ | ◐ | ◔ | |
| (Iwasaki et al. 1993) | | ● | | ◐ | | | ● |
| (Snavely & Papalambros 1993) | ◐ | ◐ | ◐ | | | | |
| (Erens & McKay 1994) | | ◐ | ◐ | | ◐ | | |
| (Gruber 1995) | | ● | | | ● | | |
| (Mizoguchi et al. 1995b) | ● | | | ◐ | | ● | |
| (Borst et al. 1997) | | ◐ | ◐ | ◐ | ◐ | | ● |
| (Guarino 1997) | ● | | | ● | ● | | |
| (Davenport & Prusak 1998) | ◔ | ◔ | | | ◔ | ◐ | ◔ |
| (Soininen et al. 1998) | ◐ | ◐ | ◐ | ◐ | ◐ | | |
| (Studer, Benjamins, & Fensel 1998) | ◐ | | | ● | ◐ | | ◐ |
| (Schreiber, Akkermans, Anjewierden, Hoog, Shadbolt, van de Velde, & Wielinga 2000) | ● | ● | ◐ | ● | ● | ● | ● |
| (Walsham 2001) | ◐ | | | | | ◐ | |
| (Sanchez & Collins 2001) | ◐ | ◐ | ◐ | | ◐ | | |
| (Mili et al. 2001) | ● | ● | ● | | ◐ | | |
| (Aler et al. 2002) | ◐ | ● | | | | | ◐ |
| (Salustri 2002) | | ◐ | ◐ | ◔ | ● | | |
| (Hicks 2003) | ◐ | | | | ◐ | | |
| (Gomez-Perez, Fernandez-Lopez, & Corcho 2004) | ● | ◐ | ◐ | ● | ● | ● | ◐ |
| (Kitamura & Mizoguchi 2004a) | | ● | ◐ | ● | ● | | |
| (Van Wie et al. 2005) | | ● | | ● | ● | | ● |
| (Power & Bahri 2005) | | ◐ | ◐ | | ◐ | | |
| (Forza & Salvador 2007) | ◐ | ◐ | ◐ | | ◔ | | ◔ |
| (Hvam, Mortensen, & Riis 2007) | ◐ | ◐ | ◐ | | ◔ | | ◐ |

A single article providing an introduction to knowledge engineering is *Knowledge Engineering: Principles and Methods* by (Studer, Benjamins, & Fensel 1998), and a quick overview of knowledge management is given in *Knowledge Management: An Introduction and Perspective* by (Wiig 1997). A single book that can introduce us to

knowledge engineering is *Knowledge Engineering and Management* by (Schreiber, Akkermans, Anjewierden, Hoog, Shadbolt, van de Velde, & Wielinga 2000).

All things and the systems they belong to are designed (at least in our minds) by a designer, and the act of designing technical things is often called *Engineering Design*. This field offers a different view on what abstractions of the world, both things and systems, should be like.

## 4.3 ENGINEERING DESIGN




The art (or science) of designing in the engineering domain is another way to describe the problem at hand. This discipline is a new science and many would even argue that it is not very scientific at all. It deals with the transformation that designers (in a very broad sense) have to create in order to realize artefacts that serve a purpose – in other words, artefacts that fulfil some need. This discipline is therefore "forced" to look at the whole and deal with requirements, needs, functions and physical objects. The literature review presented here covers engineering design and such synonyms as product design and mechanical design.

As the suggested modelling technique can be viewed as part of a design process in which it can be considered a tool, it is beneficial to investigate that aspect as well as what has been done to formalize engineering design.

### 4.3.1 WHAT IS ENGINEERING DESIGN?

Humans have been doing engineering design for thousands of years. For almost all of that time, engineering design has been done intuitively. Skilled people who mastered the design and were also able to construct the objects were responsible. Teaching was carried out in the master-apprentice style and was applied with great success. It is only in recent times that this system has not sufficed. There are two main reasons for this: complexity and time – the complexity of society, needs, knowledge, technology and construction, and time for marketing. So, the need for a teachable, systematic approach to engineering design is growing greater and greater, and along with this, our need to understand the design process. This thesis leans on the last several decades of engineering design research, mainly from Europe and USA, but also from Japan. The Accreditation Board for Engineering and Technology (ABET, www.abet.org) defines engineering design as:

> *... the process of devising a system, component or process to meet desired needs. It is a decision-making process (often iterative), in which the basic sciences, mathematics, and engineering sciences are applied to convert resources optimally to meet a stated objective. Among the fundamental elements of the design process are the establishment of objectives and criteria, synthesis, analysis, construction, testing, and evaluation. (ABET, www.abet.org)*

The "bible" on engineering design must be considered to be Pahl and Beitz's phenomenal work, *Engineering Design: A systematic approach,* which was first published in German in the late 1970s and is now in its third English incarnation (Pahl et al. 2007). It tries to cover all aspects of designing in a comprehensible way and in a precise German manner. Pahl and Beitz formulate their view of an engineering design definition in plain and straightforward text:

> *The main task of engineers is to apply their scientific and engineering knowledge to the solution of technical problems, and then to optimize those solutions within the requirements and constraints set by material, technological, economic, legal, environmental and human-related considerations.*
>
> *(p.1) in (Pahl, Beitz, Feldhusen, & Grote 2007)*

Another important European school is the work of Hubka (and Eder), which defines the Technical System (Hubka & Eder 1987), and their later, more design-oriented handbooks, such as "Design Science" (Hubka & Eder 1996). Andreasen's work should not be forgotten in this respect; his "Domain Theory" and the work with Hubka is of great significance. The theory of technical systems has been illustrated (Figure 36), and we go more into Andreasen's work in the next chapter. On the other side of the pond, there are two major players (among others), product development as suggested by (Ulrich & Eppinger 2004), and the axiomatic approach preached by (Suh 1998). Let us look at selected works on engineering design and the most relevant concepts they present.

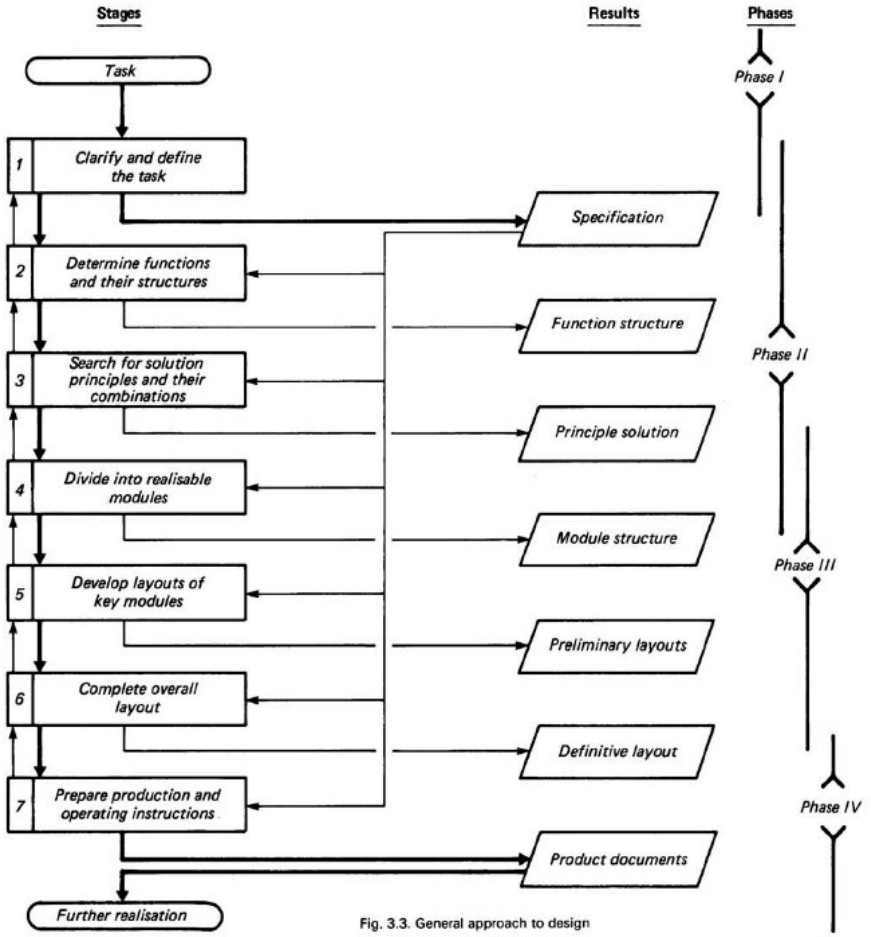



**Figure 47 - General approach to design (VDI 1986)**

### 4.3.2 LITERATURE ON ENGINEERING DESIGN

Like the literature on knowledge engineering, the most important aspects drawn from engineering design are the multi-abstraction levels and relations. The authors have very different ways of dealing with this subject. The first to be mentioned is the procedural approach of the German school, here as presented in the VDI standard of the guild of German engineers; see Figure 47.

This is largely based on Pahl and Beitz's work and is also discussed in their book (Pahl, Beitz, Feldhusen, & Grote 2007). It is "linear" in its approach, even though it has feedback arrows. It is especially good for educational purposes because of its pedagogical nature. The multi-domain approach (Andreasen 1991) is not linear and is as "get-to-work"-like as the German one. It is based on Hubka's concepts and highlights the fact that designers work in four domains when designing, as shown in Figure 48.

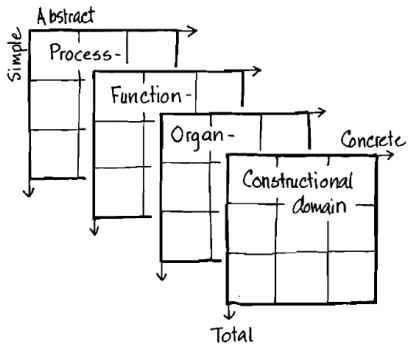



**Figure 48 - The four domains in Domain Theory (Andreasen 1991)**

Transformation of functions to structure is the focus of Gero's approach to the design process, as seen in Figure 49.

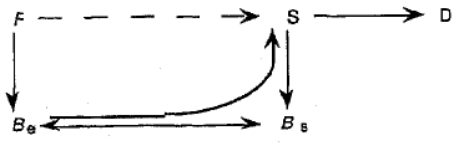



**Figure 49 - Design process (Gero 1990)**

> *Thus, design is purposeful, and the activity of designing is goal oriented. The metagoal of design is to transform requirements, generally termed functions, which embody the expectations of the purpose of the resulting artefact, into design descriptions.*
>
> *(p.28) in (Gero 1990)*

Note that the key attribute of this view is to show how knowledge "disappears" after the design is completed. What is noteworthy in both Gero's and Andreasen's approach is that needs or requirements are not included in the process. It is assumed in both cases that the mapping of those to functions or processes has been accomplished. Pahl and Beitz use a lot of time explaining how to do this, and so does MIT's Suh in his axiom approach (Suh 1998). He has four aspects and mapping between them is the central concept in his approach, as shown in Figure 50. Although the axiomatic design is an elegant theory, mathematically, the author is less concerned with explaining how to populate the different domains. Apart from that, Suh's theory contains some excellent thoughts, like those stated in his axioms and corollaries, and they have served

as inspiration in this thesis; see Suh's book, *The Principles of Designs,* for details (Suh 1990).

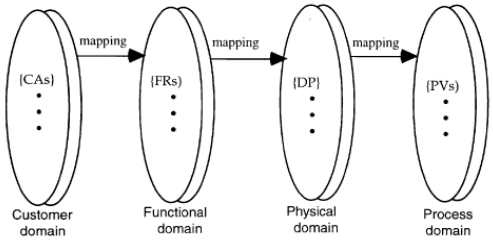



**Figure 50 - Axiomatic design and the four domains (Suh 1998)**

On the other end of the design theory spectrum, along with Pahl and Beitz, is Steward's design structure matrix (DSM) approach. Its tool-like appearance and clear guidelines aid in structuring what the other approaches do less well. Let us now list and compare key aspects of these theories in Table 12.

**Table 12 - Selected literature on engineering design**

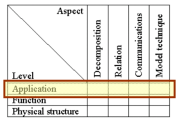
| Author \ Levels & Aspects | Application | Function | Artefact | Decomposition | Relation | Communications | Model technique |
|---|---|---|---|---|---|---|---|
| (Steward 1981), | | ◔ | ◔ | ● | ● | | ● |
| VDI 2221 (VDI 1986) | ◐ | ● | ● | ◐ | ◐ | | |
| (Hubka & Eder 1987) | | ● | | ◔ | | | ◐ |
| (Gero 1990) | ◔ | ◐ | ◐ | ◔ | ◐ | | ◔ |
| (Andreasen 1991) | ◔ | ◐ | ◐ | ◔ | ◔ | | |
| (Uschold & Gruninger 1996) | ● | | | | ◐ | ● | |
| (Hubka & Eder 1996) | ◐ | ◐ | ◔ | ◔ | ◐ | ◔ | ◐ |
| (Suh 1998) | ◐ | ◐ | ◐ | ◐ | ◐ | | ◐ |
| (Buede 2000) | ● | ● | | ● | ● | ◐ | ● |
| (Martin & Ishii 2002) | | ● | ● | ◔ | ◔ | ◔ | ◐ |
| (Lossack 2002) | ◐ | ◐ | ◐ | ◔ | ◔ | | ◔ |
| (Sim & Duffy 2003) | ◐ | ◐ | ◐ | ◐ | ◐ | ◐ | ◐ |
| (Alonso-Rasgado, Abdelkafi, Thompson, & Elfstrom 2004), | ◐ | ◐ | ◐ | ◔ | ◔ | | |
| (Ulrich & Eppinger 2004) | ● | ● | ● | ● | ◐ | | ◔ |
| (Pahl, Beitz, Feldhusen, & Grote 2007) | ● | ● | ● | ● | ● | | ◐ |
| (Kusiak & Salustri 2007) | ◔ | ◐ | ◔ | | ◔ | ◐ | |

There is quite an even spread in Table 12. Most works deal with most aspects. But the same holds true here as in earlier chapters – no single author (or only a few) can be used directly. A merger of almost all aspects is needed to construct a viable view on design. The authors used here present quite different views of the design process. A single reference seems to tackle almost all aspects – *The Engineering Design of Systems* by (Buede 2000). It is a fine summary and deals with relations in depth, but it is very high-level and silo-like, since it does not suggest mating all aspects. For a

summary of all views, the best bet would be (Sim & Duffy 2003). Two major books that include artefact structure are (Ulrich & Eppinger 2004) for a lighter read and (Pahl, Beitz, Feldhusen, & Grote 2007) for a more thorough engineering view.

This concludes the review of literature that includes holistic views. Now, it is time to look at the theory from each level and aspect, and to see how it adds to the necessary understanding.

## 4.4 APPLICATION




The first level to look at is the *Application* level. The term application may be a little confusing, but the author can find no other single term to replace it. This level is about the purpose of things, how they are applied in a specific context to solve a specific problem. So this is about the customer view of things, their requirements or needs, the purpose of objects, the effect required from the designed object, and then the task or problem-solving processes involved.

### 4.4.1 WHAT IS APPLICATION?

Application is about context. It could also be called needs, requirements, tasks, process, purpose etc. The term application is select because the case company uses it, and no other term has been identified that better captures the nature of this abstraction. It can be a little confusing, since the term has different meanings in different domains. It is about customer needs and how they are tied to function groups or applications. One aspect of application is the environment in which the object finds itself, and another involves the underlying functions and physical structures that supply the solution to the problem or the need that the application is fulfilling. From that aspect this could maybe be called "Service" and defined as a group of basic functions.

### 4.4.2 LITERATURE ON APPLICATION

The application is surely about context, but it is also about goal. In the spirit of designing artefacts, we assume that objects have a purpose and that it has to do with solving some "problem" or need. Extensive literature on problem solving can be drawn upon. Problem-solving methods (or PSM) seem to follow a generic architecture, like the one presented in Figure 51.

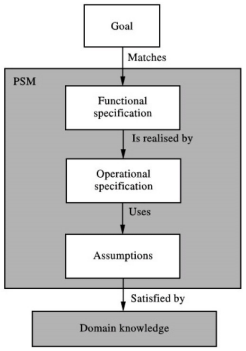



**Figure 51 - The architecture of a PSM (Teije et al. 1998)**

These PSMs are one way to deal with the job of solving problems.

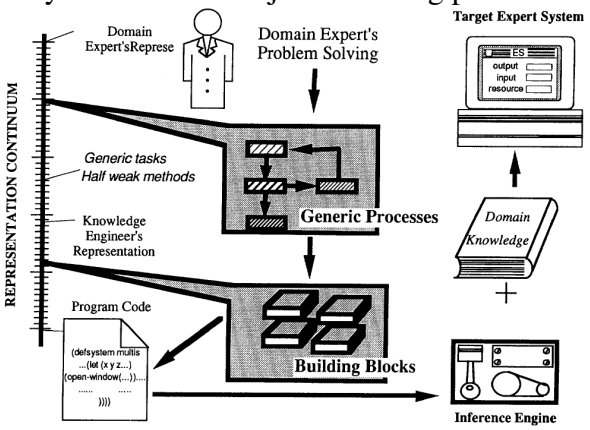



**Figure 52 - Abstraction level of generic processes and building blocks (Mizoguchi et al. 1995a)**

Another way is tasks. Task ontology, as the one suggested by (Mizoguchi, Tijerino, & Ikeda 1995a), introduces complete sets of verbs and nouns to describe tasks, and then uses a representation continuum to place the knowledge needed, as presented in Figure 52.

The knowledge view of tasks is well defined:

> *... a task structure... lays out the relations between a task, applicable methods for it, the knowledge requirements for the methods, and the subtasks set up by them.*
>
> *(p.61) in (Chandrasekaran 1990)*

The propose-critique-modify family of methods by (Chandrasekaran 1990) reflect a work process used everywhere, the *classical* trial and error process. A task structure can also have an action view instead of the knowledge view presented earlier. An action task structure for diagnosis is presented in Figure 53. Here, the structure does not reflect the knowledge needed.

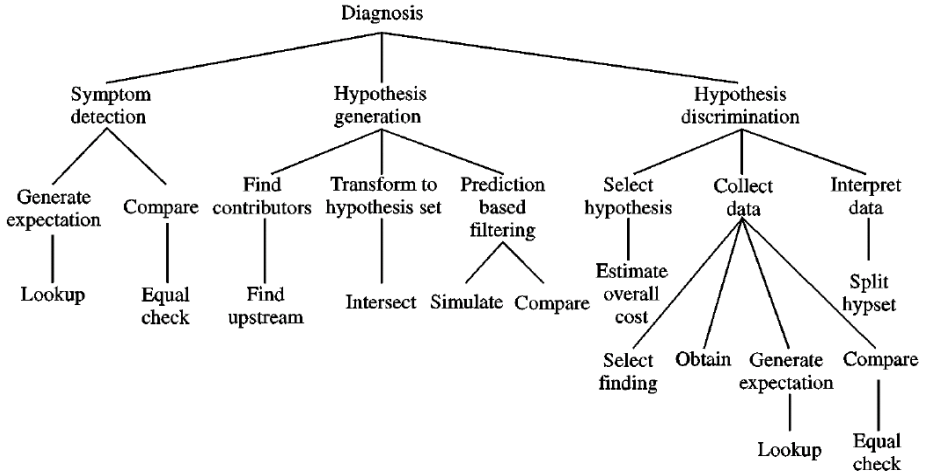



**Figure 53 - Task structure for diagnosis (Orsvarn 1998)**

Solving a problem with searches is also quite common. In this thesis, this is quite relevant since the suggested solution uses this kind of technique to reduce the complexity of the problem at hand. A visualization of such a search problem in configuration can be seen in Figure 54.

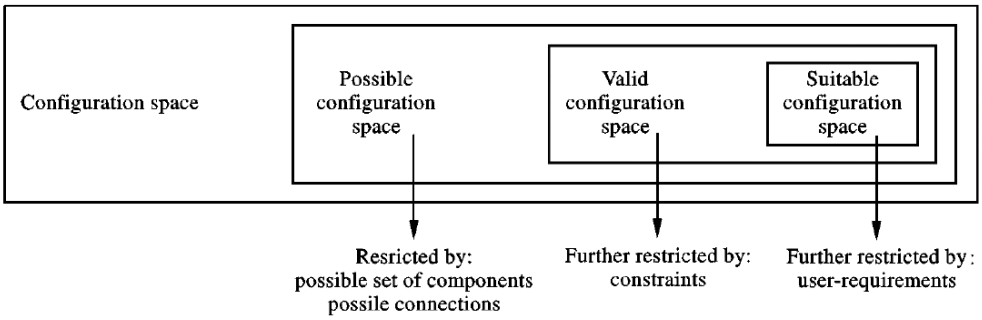



**Figure 54 - Configuration task as search problem (Teije, Harmelen, Schreiber, & Wielinga 1998)**

The search is not random but builds on some requirements or needs. So, requirements are very important and play a major role in constructing the applications. A fine way to show the impact of requirements on the modelling of objects is shown in Figure 55.

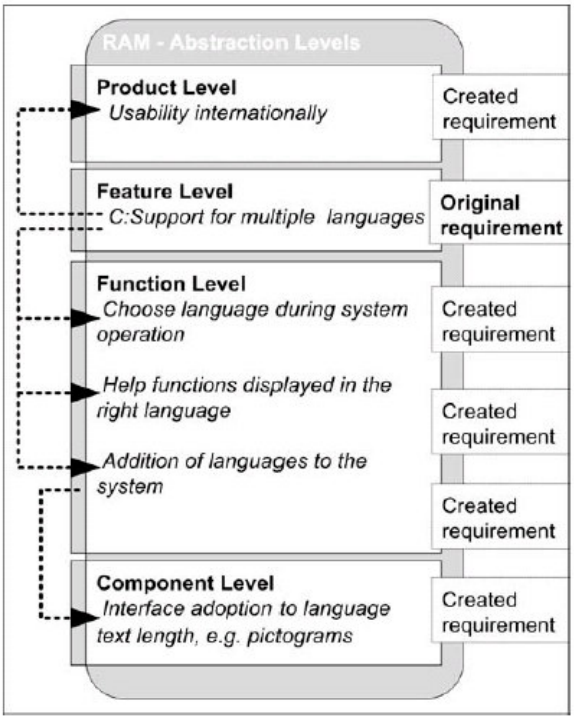



**Figure 55 - Impact of requirements (Gorschek & Wohlin 2006)**

Requirements and their capture is a difficult process. Some guidelines on how to do this can be found in (Arthur & Gröner 2005). They state that we have to look at several factors when listing requirements, such as indoctrination, preparation, elicitation and evaluation. The framework is shown in Figure 56.

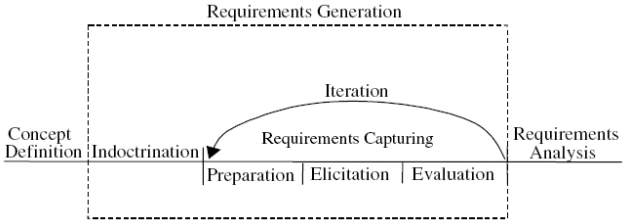



**Figure 56 - The requirement generation framework (Arthur & Gröner 2005)**

Two things are noteworthy in this framework, indoctrination and elicitation. The former is about "educating" both engineers and customers about what the problem is,

what the needs are, and how this can be described. The latter is about stating the requirements, because requirements can be implied by the customer as well as stated explicitly. Catching all the necessary requirements can be quite tricky. Modelling requirements and their evaluation are also very important. Categories for sorting requirements and identifying how to test them are presented in Figure 57.

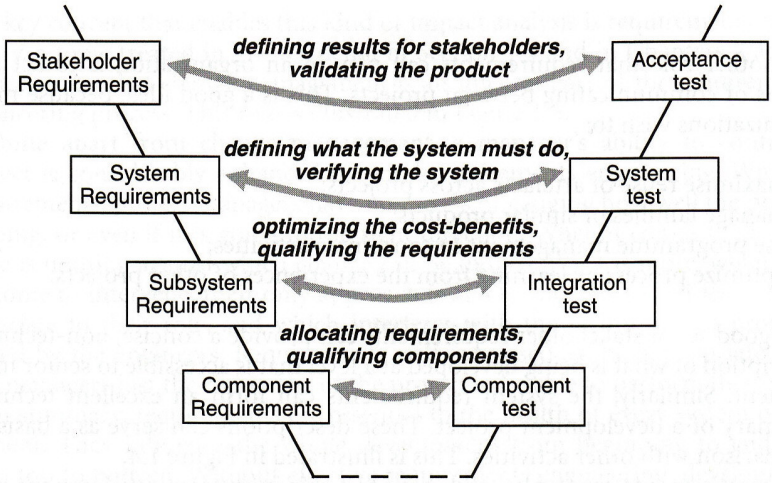



**Figure 57 - Requirement engineering in layers (Hull, Jackson, & Dick 2005)**

This "rough" sorting should aid in making an overall model of applications and could even supply some knowledge to other levels. Coupling requirements to specific models is shown in Figure 58.

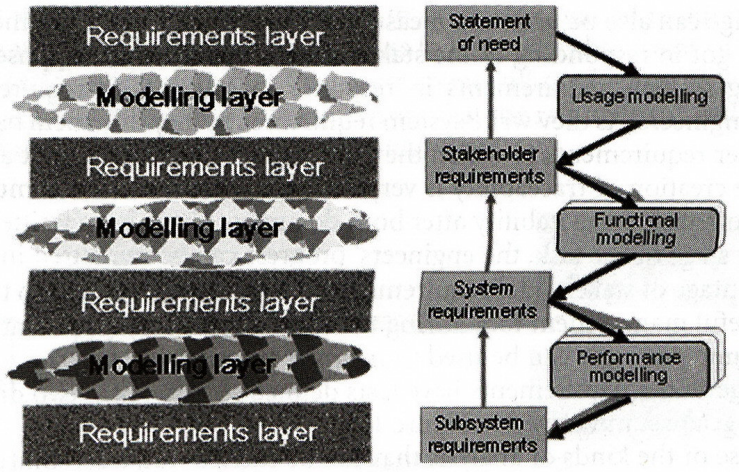



**Figure 58 - Requirements and modelling (Hull, Jackson, & Dick 2005)**

Keeping these aspects in mind when constructing an application view on objects should make things a little more structured than before. The resulting structure should have general trends that can be used to generate generic application models. A selection of literature that deals with application can be seen in Table 13.

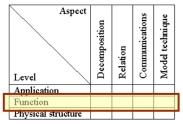
The authors and work referred to in Table 13 have different views on structuring applications. They all deal with the purpose of things in their own way but with quite different approaches, from scenarios, tasks and processes to intelligent agents, requirement engineering and goal orientation. Also here, no single author captures all the concepts necessary to adequately describe applications, but a merger from many could guide the way. For a single article on articulation of requirements and its use in application, see *Requirements Abstraction Model* by (Gorschek & Wohlin 2006) and a book on the same subject, *Requirements Engineering,* by (Hull, Jackson, & Dick 2005).

**Table 13 - Selected literature on applications**

| Author \ Levels & Aspects | Decomposition | Relation | Communications | Model technique |
|---|---|---|---|---|
| (Cohen & Lee 1990) | ◔ | ◐ | ◔ | |
| (Chandrasekaran 1990) | ● | ◔ | ◔ | |
| (Mizoguchi, Tijerino, & Ikeda 1995a) | ● | ● | ◔ | ◐ |
| (Wooldridge & Jennings 1995) | | ● | ● | ◐ |
| (Uschold 1996) | | ◐ | ◔ | ● |
| (Patnaik & Becker 1999) | ◔ | | ◐ | |
| (Kavakli 2002) | ◐ | ◐ | | ● |
| (Dutoit & Paech 2002) | ◔ | ◐ | ◐ | |
| (Buhne et al. 2005) | ◔ | ◔ | ◔ | ● |
| (Leite et al. 2005) | ◐ | ◐ | | ● |
| (Arthur & Gröner 2005) | ◔ | ◔ | ● | |
| (Hull, Jackson, & Dick 2005) | ● | ◐ | ● | ◐ |
| (Zhang et al. 2006) | | ● | | ● |
| (Gorschek & Wohlin 2006) | ● | ● | ● | ◐ |

Applications are realized with some functionality of some components. The next level, functions, is a very important level. The next section is about functions – what they are, how to describe them, and finally, how to use them in context with the other levels of abstractions.

## 4.5 FUNCTION




Function or functionality is a tricky thing. We all understand it, but then again we do not. Intuition tells us how to understand it, but it is usually in a specific context. In this thesis, the premise is that artefacts can be described on different abstraction levels, and the function level is the level above the actual physical components. It is therefore important to decide what is function and formulate its role in the greater scheme. A lot of literature exists on functional modelling, but strangely enough, the author could not find a single literature review of functional modelling.

In this chapter, we discuss the concept of function, decide how to use it, and reflect on the literature considered relevant for this endeavour. But first, what is function?

### 4.5.1 WHAT IS FUNCTION?

Although designers agree that function is related to behaviour, two distinctive differences contrast the definitions of function.

1. Scope of function:

a. Function is just a part of behaviour (abstracted behaviour).
b. It also includes the designer's purpose or intention.

2. Behavioural representation that forms the foundation of functional reasoning, with such functions as:
   a. State transitions
   b. Input/output relationship – the most common view
   c. Physical phenomena
   d. A mixture of state transitions and physical phenomena.

The major factor for these differences is the application domain. Function representation based on the input/output relationship has trouble representing a function that does not transform anything. The state-transition-based representation describes the physical principles that cause the state transition less explicitly than does the physical-phenomenon-based representation. (Umeda & Tomiyama 1997) conclude by putting forth an interesting research issue: Under what condition is it more natural to relate a function to an input/output relationship, to physical phenomena, or to a state transition? Examples of authors that take the intention view of function are Gero and Umeda. Gero connects the final goal and behaviour:

> *Function has been defined in another context as the relation between the goal of a human user and the behaviour of a system.*
> *(p.28) in (Gero 1990)*

While Umeda recognizes function as human cognition of goals:

> *A description of behaviour abstracted by humans through recognition of the behaviour in order to utilise it.*
> *(p.183) in (Umeda et al. 1990)*

This coincides with the third author, de Kleer, and his qualitative physics. He states that structure is *what the device is*; behaviour is *what a device does;* and function is *what a device is for* (de Kleer & Brown 1984). He sets out with a very ambitious goal – to develop language that is able to infer behaviour from structure. His work can be seen as basic for understanding the elements of functions and functional thinking.

The duality of function (behaviour vs. behaviour with intention) is nicely shown in Hubka's work. Remember Hubka's framework in Figure 36,, where the author tries to combine all three (a, b and c) ways to describe a function. The function of the TS or machine system (MS) is function as abstracted behaviour; the author calls this *effect*, and the function with intention is the activity, which is the process that transforms input to output. So, function can be a verb-noun description of the MS or activity that serves some purpose; this is shown in Figure 59.

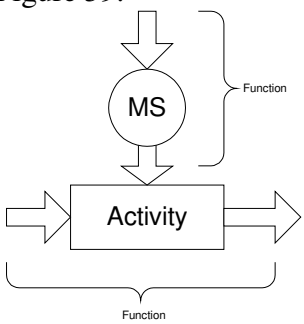



**Figure 59 - The two views on function - activity versus technical system (Andreasen 2007) based on (Hubka & Eder 1996)**

To recapitulate, two kinds of functions are shown, the function of a MS (e.g. rotating a drill) and the function of an activity (making a hole). Other authors separate functions from intentions and introduce goals instead. Lind defines function as:

*Functions are useful behaviour.* *(p.16) in (Lind 1990)*

We can thus say that he adheres to the narrower scope of function, without intentions. Lind also states that functions have the role to fulfil goals in the system, and these *must* be based or caused by certain behavioural properties of the system, and function cannot be identified without knowledge of goals, structure and behaviour of the system.

The final aspect we need to tackle is the consequence of design. Some function may contribute to achievement of some selected goals, while others may be detrimental if they prevent goal achievement. The latter are called *dysfunctions*. Dysfunctional behaviour is usually not an intended feature of man-made systems, but it often results from design constraints. Actually, the purpose of many systems is to eliminate or control dysfunctions (consider the lubrication system in a motor). When designing, the designer wants some functions, and therefore they are known to the system analyst. These are *manifest* functions, while others are not recognized – the so-called *latent* functions – but these latent functions can become useful later on. An attempt to visualize these different terms is shown in Figure 60.

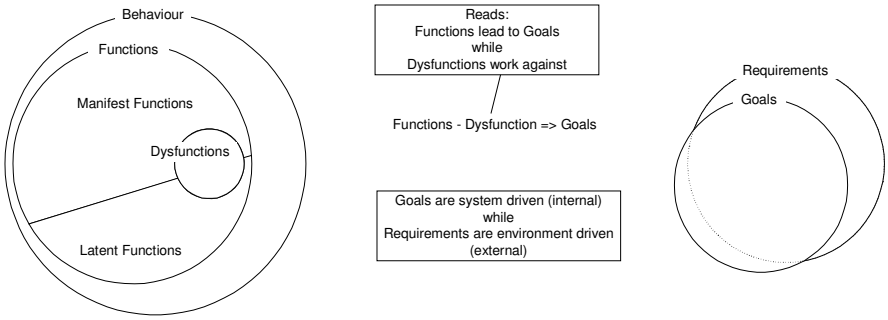



**Figure 60 - Functions – terms and goals**

Now that a picture of functions is drawn, let us look at the key concepts presented in the literature on functions and functional modelling.

### 4.5.2 LITERATURE ON FUNCTIONS

Because functions cannot be seen directly, they can be quite tricky. And even though humans have designed for millennia, explicit views on functions and functionality are quite recent. The whole field of functional modelling works on a concept introduced by (de Kleer & Brown 1984): the *no-function-in-structure principle*. It states that to be able to make any kind of reasonable function description, we have to separate the structure of artefacts from the intended functionality. In other words, we have to describe functions explicitly and not allow implicit functions to remain hidden in the artefact. Let us examine several methods for functional modelling and see how this is achieved.

The multi-level flow modelling or MFM by (Lind 1990) is a systematic way of dealing with functions. It has a two-axis definition: the *means-to-an-end* axis, which via components-functions-goal can be viewed at different decomposition levels, and the *whole-to-part* axis, as shown in Figure 61.

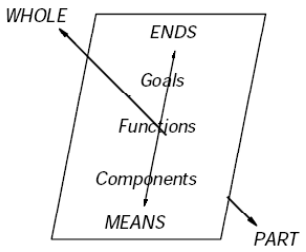



**Figure 61 - Holistic approach to functions (Lind 1990)**

Lind goes on to define a systemic view that has five basic categories:

1. a *goal* (objective) or set of goals
2. alternative *means* that can be used to reach the goal(s) and are sets of system *elements* or *strageties*
3. *resource* expenditure on the system, including a quantification of the amount of resources spent to achieve goals (time, material, energy, knowledge, personnel etc.)
4. a mathematical model or logical *model,* i.e. a representation of *relations* between goals, alternative means, environment and requirements imposed on resources
5. *criteria* by which the preferred alternatives are selected

Lind's work is very complete and is one of the best on functional thinking in design. He goes through all aspects in his holistic model (Figure 61) and explains or defines each aspect. We will not go through all his arguments but only point out essential ones and refer the reader to look at Lind's work for the rest.

It is not possible to define all behaviours of a system, because then it would be necessary to put it into all possible contexts. The problem of context dependency of functional ascriptions appears sometimes as a problem of categorization of the functional properties of a system. The identification of functionality depends on both decomposition level and context. When attempting to ascribe goals to systems, it is assumed that the behaviour of the system is directed towards certain ends. Man-made systems are purposeful and their behaviour, when they behave properly, is directed towards the achievement of goals. Goals are not always easy to identify, because they are not always explicitly given or represented in the system.

There are two kinds of systems when it comes to goals, the *goal-oriented system* and the *goal-controlled system*:

- **Goal oriented**: When the behaviour is directed towards a goal but is not controlled by the goal (information about success of goal achievement is not used in control of the system). In low change, high predictability systems, this is usually sufficient to ensure success.
- **Goal controlled**: When goals are used actively to control the system. In the case when environment or goals are changing, goal achievement is used by decision agents to control the system. This is the feedback principle.

This distinction between goal-oriented and goal-controlled systems is important when modelling artefacts and attention should be paid to proper identification of system goals. And then there are requirements. It is important to make a clear distinction between requirements and goals. Requirements are placed on systems from the outside world as conditions for successful adaptation; however, a system may have goals that are different from the requirements that come from the outside – e.g. goals like safety, production, or economy.

On the other axis, the means-end relations can be divided into two main categories: *achievement relations* and *condition relations*. Achievement relations then have two sub-categories: achieve relations and achieve-by-control relations (Lind 1994).

Many authors have taken the *state* view of functions. This states, in brief, that functions are behaviours that influence states of systems. One of the main advocates of Function-Behaviour-State (FBS) modelling is Yasushi Umeda. His FBS modeller is about this transition and a visualization of its relations is shown in Figure 62.

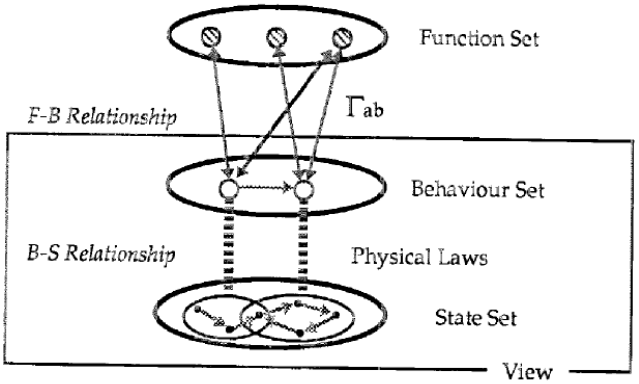



**Figure 62 - Relationships between Functions, Behaviours and State (Umeda, Takeda, Tomiyama, & Yoshikawa 1990)**

The FBS modeller declares that each function is an instance of a function prototype, as defined in this triplet (Umeda & Tomiyama 1997):

| | |
|---|---|
| *Name*: | Symbol representing the designer's intention in the form of "to do something." |
| *Decomposition*: | Networks of sub-functions. |
| *F-B relationship*: | Physical features. |

The authors say that FBS modeller can aid in: *Specification of the required functions*, *Functional decomposition*, *Embodiment of the functions*, *Construction of the behaviour network* and *Behaviour simulation and evaluation*. Since the FBS modeller builds on function prototypes or functional elements, it is appropriate to mention a few more of them.

Since the beginning of functional modelling, authors have been wondering about the *building blocks* of functions. The obvious place to start is with (Pahl, Beitz, Feldhusen, & Grote 2007). They deal with *flows* of different kinds and try to systemize the use of language and diagrams. Lind takes this work further and suggests a vocabulary for describing functions (Lind 1990). Most other authors also suggest some standardization of building blocks, but the most complete work in this area is that of Stone and his colleague at University of Missouri-Rolla (Stone & Wood 2000).

All of these authors work on the premise that functions can be described using *verb-noun* combinations, a legacy from *Value Engineering* that started in the 1960s. This is actually almost universally agreed upon in the functional modelling field. Stone then used the work of his predecessors and set out to find all verbs and nouns and construct a complete language. His result is the *Functional Basis* (Stone 1997), where he defines *flow classes* and *function* classes corresponding to the noun and verb (in that order).

Sub-sets of his two classes are shown in Figure 63 and Figure 64. Stone also introduced a graphical notation in his work, which can be seen further on in Figure 94.

| Class | Basic | Sub-basic | Complements | |
|---|---|---|---|---|
| Material | Human | | Hand, foot, head ,etc. | |
| | Gas | | | |
| | Liquid | | | |
| | Solid | | | |
| Signal | Status | Auditory | Tone, Verbal | |
| | | Olfactory | | |
| | | Tactile | Temperature, Pressure, Roughness | |
| | | Taste | | |
| | | Visual | Position, Displacement | |
| | Control | | | |
| | | | Bond graph based complement | |
| Class | Basic | Sub-basic | Effort analogy | Flow analogy |
| Energy | Human | | Force | Motion |
| | Acoustic | | Pressure | Particle velocity |
| | Biological | | Pressure | Volumetric flow |
| | Chemical | | Affinity | Reaction rate |
| | Electrical | | Electromotive force | Current |
| | Electromagnetic | Optical | Intensity | Velocity |
| | | Solar | Intensity | Velocity |
| | Hydraulic | | Pressure | Volumetric flow |
| | Magnetic | | Magnetomotive force | $\frac{d}{dt}$ magnetic flux |
| | Mechanical | Rotational | Torque | Angular velocity |
| | | Translational | Force | Linear velocity |
| | | Vibrational | Amplitude | Frequency |
| | Pneumatic | | Pressure | Mass flow |
| | Radioactive | | Intensity | Decay rate |
| | Thermal | | Temperature | Heat flow |
| Overall increasing degree of specification ➡ | | | | |

**Figure 63 - Flow class (Stone 1997)**

| Class | Basic | Flow restricted | Synonyms |
|---|---|---|---|
| Branch | Separate | | Switch, Divide, Release, Detach, Disconnect, Disassemble, Subtract, |
| | | Remove | Cut, Polish, Sand, Drill, Lathe |
| | Refine | | Purify, Strain, Filter, Percolate, Clear |
| | Distribute | | Diverge, Scatter, Disperse, Diffuse, Empty Absorb, Dampen, Dispel, Resist, Dissipate |
| Channel | Import | | Input, Receive, *Allow*, Form Entrance, *Capture* |
| | Export | | Discharge, Eject, Dispose, Remove |
| | Transfer | | |
| | | Transport | Lift, Move |
| | | Transmit | Conduct, Convey |
| | Guide | | Direct, Straighten, Steer |
| | | Translate | |
| | | Rotate | Turn, Spin |
| | | Allow DOF | Constrain, Unlock |
| Connect | Couple | | Join, Assemble, *Attach* |
| | Mix | | Combine, Blend, Add, Pack, Coalesce |
| Control Magnitude | Actuate | | Start, Initiate |
| | Regulate | | Control, *Allow*, *Prevent*, Enable/Disable, Limit, Interrupt, Valve |
| | Change | | Increase, Decrease, Amplify, Reduce, Magnify, Normalize, Multiply, Scale, Rectify, Adjust |
| | | Form | Compact, Crush, Shape, Compress, Pierce |
| | | Condition | |
| Convert | Convert | | Transform, Liquefy, Solidify, Evaporate, Condense, Integrate, Differentiate, Process |
| Provision | Store | | Contain, Collect, Reserve, *Capture* |
| | Supply | | Fill, Provide, Replenish, Expose |
| | Extract | | |
| Signal | Sense | | Perceive, Recognize, Discern, Check, Locate |
| | Indicate | | Mark |
| | Display | | |
| | Measure | | Calculate |
| | Stop | | Inhibit |

**Figure 64 - Function class (Stone & Wood 2000)**

Another framework for dealing with functional thinking is the *Goal Tree – Success Tree* (GTST) by (Modarres & Cheon 1999). GTST modelling is a functional decomposition framework to describe and model complex physical systems in terms of *objects*, *relationships*, and *qualities*. This framework differs from Lind's MFM by unifying the two axes into one, as shown in Figure 65. It still deals with means-end and part-whole, but the modelling approach is just a little different.

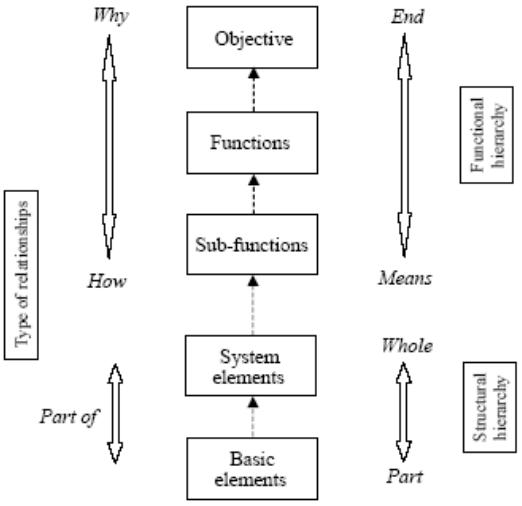



**Figure 65 - The GTST framework (Modarres 1993)**

The author tries to incorporate several aspects into a functional hierarchy. Their overall function-centred system description is shown in Figure 66. What this work augments in relation to the work of Lind is the strong focus on how to deal with relations between aspects.

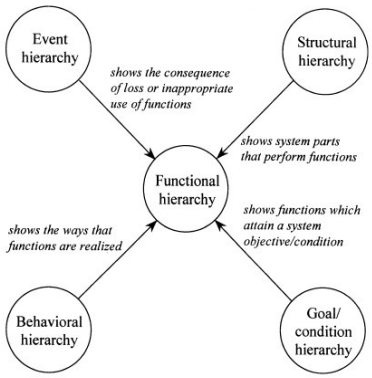



**Figure 66 – Function-centred hierarchy (Modarres & Cheon 1999)**

The authors try to suggest ways to map all five aspects in Figure 66 in a comprehensive way. They try to show this visually in Figure 67.

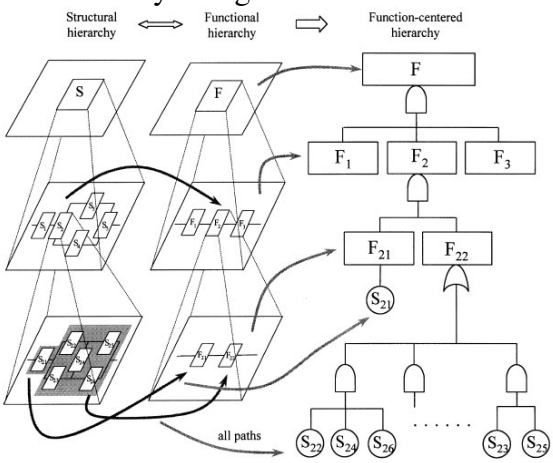



**Figure 67 - Function-centred hierarchy with mappings (Modarres & Cheon 1999)**

To handle complexity and, not least, to show relationships, two GTST are placed on the x-y axis matrix structure to map one to the other. This is a good attempt to join different

types of modelling techniques for a greater presentation of the problem at hand (more on this in the section *Modelling techniques* on p.103). This visualization of relations is shown in Figure 68.

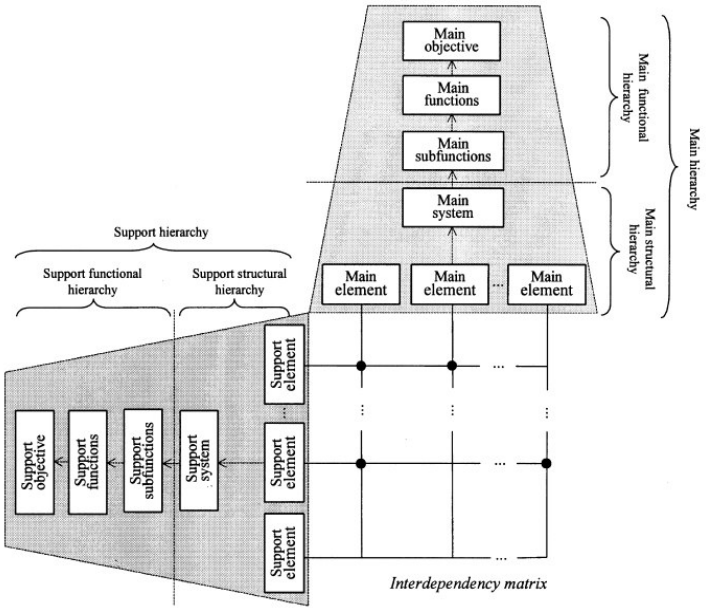



**Figure 68 - Relations between GTST structures (Modarres & Cheon 1999)**

The basic thinking in Figure 68 is very much along the lines of the premises in this thesis. One of the main advantages of the interdependency matrix is the ability to track the effect of a missing element on the overall supply of functions.

A further development or twist on MFM is the *Goal function modelling* (GFM) by (Soerensen 1999). This method is developed particularly for reuse-based design of complex industrial control systems. It focuses on the identification and organization of different kinds of knowledge, but mainly for control systems. The strength of GFM is its why, what and how, as shown in Figure 69.

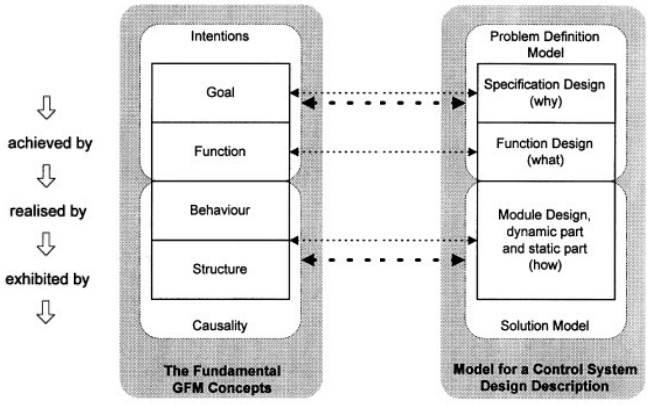



**Figure 69 - GFM concepts (Soerensen 1999).**

We can see that functions and behaviour are tightly coupled. An excellent work on functional view is the work of (Kitamura & Mizoguchi 2004a) on the *Ontology-based systematization of functional knowledge*. In this work, the authors offer a look at the philosophical and fundamental issues of functional modelling.

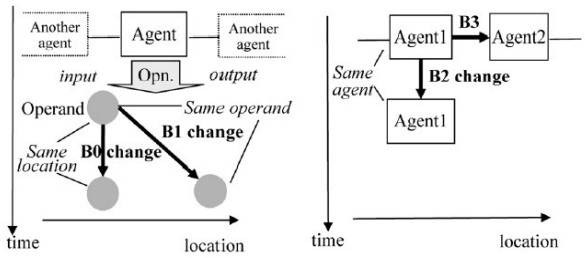



**Figure 70 - Four definitions of behaviours (Kitamura & Mizoguchi 2004a)**

The authors identify four kinds of behaviour. The behaviour definitions are illustrated in simplified situations in Figure 70 and are as follows:

*B0 behaviour is defined as the change of an attribute value of an operand at the same location over time.*
*B1 behaviour is defined as the change of an attribute value of an operand from that at the input port of a device to that at the output of the device.*
*B2 behaviour is defined as the change of something inside of a device rather than input/output ports. The 'something' could be motion of a part of the device or inner state of the device.*
*B3 behaviour is defined as any behaviour to another device. The important aspect here is B0 and B1 behaviours are concerned with operands rather than devices.*
*(p.335) in (Kitamura & Mizoguchi 2004a)*

The authors also suggest a step-by-step method to decompose functions, a very "German-engineering-like" method, almost straight out of Pahl & Beitz. Kitamuras and Mizoguchi's method for functional decomposition is shown in Figure 71.

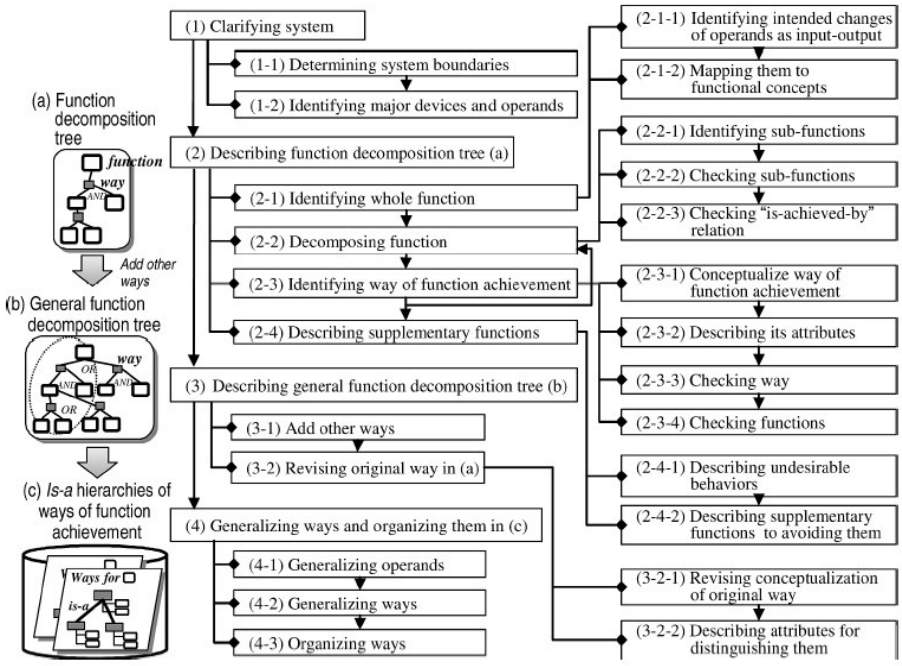



**Figure 71 - Functional decomposition (Kitamura & Mizoguchi 2004b)**

This categorization of behaviour could aid in sorting functions and indicate how to structure functional views of artefacts. In another article, the same authors have suggested a guide on how to go about decomposing the functionality of an artefact into

a functional hierarchy. In Figure 71, the authors highlight four steps to go through to generate functional decomposition. The authors give us pointers on how to tackle the construction. Their guide is given in **Table 14**.

**Table 14 - Guidelines for function decomposition (Kitamura & Mizoguchi 2004b).**

| | | |
|---|---|---|
| F: About functions and behaviours | | |
| F1 | | A function represents "what to achieve" only and does not imply "how to achieve". |
| | 1-1 | A device is a black-box. The inside is not shown at a level. |
| F2 | | A function represents (a teleological interpretation of) changes in physical things within the system boundary. |
| | 2-1 | Do not describe the designer's activities. |
| | 2-2 | Distinguish product's functions, manufacturing processes, and recycling activities. |
| | 2-3 | Determine a system boundary with a pre- and post-process. |
| F3 | | Agent of functions should be a "device" in the physical world. |
| | 3-1 | A human operator can be regarded as a "device". |
| | 3-2 | Designers and manufacturer should be distinguished. |
| | 3-3 | Sizes of devices decrease in function decomposition. |
| | 3-4 | A device can be virtual and dynamic. |
| F4 | | Decompose functions which imply kinds of operands and/or degrees of results for functions. |
| | 4-1 | Such implications are represented as attributes of ways of function achievement. |
| S: About relations between sub-functions | | |
| S1 | | Identify states of operands that flow to sub-functions. |
| S2 | | Time passes along this relation. |
| S3 | | Roles of things as operands should not be changed in a series of functions. |
| A: About "is-achieved-by" relation and way of function achievement | | |
| A1 | | The "is-achieved-by" relation represents aggregation. |
| | 1-1 | The total changes in sub-functions should correspond to changes in the whole function. |
| | 1-2 | This relation does not imply a time interval. |
| | 1-3 | This relation is not an "is-a" relation. |
| A2 | | A sub-function should explicitly contribute to a macro-function. |
| | 2-1 | Explicate implicit sub-functions. |
| A3 | | The way of function achievement represents a single principle. |
| | 3-1 | Decompose compound principles. |
| | 3-2 | Distinguish them from other ways at the principle level. |
| | 3-3 | If possible, conceptualize neither tools nor operands but principles. |
| | 3-4 | A way should refer to a direct macro-function. |
| A4 | | Distinguish supplementary functions from essential functions. |

An evaluation of functional literature now follows in Table 15. Here, the focus is not only on whether the authors deal with the four aspects but also whether they deal with behaviour, states and goals as well.

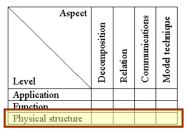
Literature on function is a little defuse and not a complete whole. In recent years, the literature shows a clear convergence. For a single article to read on functional thinking, there are two that should be highlighted as best representing the goal this thesis is trying to achieve, *Function-centred modelling of engineering systems using the goal tree-success tree technique and functional primitives* by (Modarres & Cheon 1999) and *Ontology-based systematization of functional knowledge* by (Kitamura & Mizoguchi 2004a). Both have a *holistic* approach to functions and give concrete advice on how to go about modelling.

**Table 15 - Selected literature on functions**

| Author \ Levels & Aspects | Behaviour | State | Goal / Purpose | Decomposition | Relation | Communications | Model technique |
|---|---|---|---|---|---|---|---|
| (Simon 1978) | | | ◔ | ◔ | ◔ | | |
| (Levesque 1984) | | | | | ◔ | ● | ● |
| (de Kleer & Brown 1984) | ● | ● | ◔ | ◐ | ◐ | ◐ | ● |
| (Pahl, Beitz, Feldhusen, & Grote 2007) | ◐ | | ◐ | ◐ | ◔ | | ◔ |
| (Clarke 1989) | | | | ◐ | ◐ | | |
| (Mittal & Frayman 1989) | | | ◔ | ◐ | ◐ | ◔ | |
| (Gero 1990) | ◐ | | ◔ | ◔ | ◔ | ◔ | ◔ |
| (Lind 1990) | ● | ◔ | ● | ◔ | ◐ | ● | ● |
| (Ulrich & Seering 1990) | | | ◔ | ● | ◔ | | |
| (Umeda, Takeda, Tomiyama, & Yoshikawa 1990) | ● | ● | | | ◐ | | ◐ |
| (Salustri & Venter 1992) | | | ◔ | ◔ | ◔ | ◔ | |
| (IDEF 1993) | | | | ● | ● | | ● |
| (Chandrasekaran et al. 1993) | ◐ | ◐ | ◐ | | | | ◐ |
| (Iwasaki, Fikes, Vescovi, & Chandrasekaran 1993) | ◐ | ◔ | ◔ | ◔ | ◔ | ◐ | ◐ |
| (Modarres 1993) | ◐ | | | ◐ | ● | | ◐ |
| (Qian & Gero 1996) | ● | ◔ | | ◔ | ● | | ◐ |
| (Rosenman & Gero 1996) | ● | | ● | | | | ● |
| (Umeda & Tomiyama 1997) | ◐ | ◔ | | | ◐ | | |
| (Erixon 1998) | | | ● | ◔ | ● | ● | ● |
| (Kirschman & Fadel 1998) | | | ◐ | ● | ● | | |
| (Modarres & Cheon 1999) | ● | ● | ● | ◐ | ◐ | ◐ | ● |
| (Soerensen 1999) | ● | | ● | ◔ | ● | ◐ | ◐ |
| (Stone & Wood 2000) | | | | ● | ● | | ● |
| (Bi & Zhang 2001) | | | | ● | ● | | ◐ |
| (Hirtz et al. 2002) | | | | ● | ● | | ● |
| (Kitamura & Mizoguchi 2004a) | ● | ◔ | ◔ | ● | ● | ◔ | ◔ |
| (Zhang et al. 2004) | ◐ | | ◐ | | | | ◐ |
| (Chakrabarti et al. 2005) | ◐ | | | ◐ | | ◐ | |
| (Van Wie, Bryant, Bohm, Mcadams, & Stone 2005) | | | | ● | ● | | ● |
| (Johannesson & Claesson 2005) | | | ◐ | ◐ | ◐ | | ● |

We must remember however that functions are *figments of our imaginations,* in that sense that humans have to ascribe function to artefacts, and therefore function cannot be observed directly. A move to artefact is now in order, and here we come to the only *observable* level and the only level the designer can influence directly.

## 4.6 ARTEFACT




Even thought artefact structuring in models is being done all over the place, research on the subject seems to be limited. Most research takes a specific view on things, and therefore we have several tracks to follow in this section. The theoretical groundwork of part-whole studies is combined with practical aspects from manufacturing (bill-of-material, BOM), modularity and engineering design.

We do not define artefact structuring here but let a review of the literature related to the domain reveal some common trends. The term *artefact* is chosen for a reason. This level could be called *parts* or *components* or *objects* or *things* or a number of other different names. Artefacts are made by humans to serve some purpose. Herbert Simon put forth the boundaries for science of the artificial in his book of the same name (Simon 1996), where he states:

> *Artificial things can be characterized in terms of functions, goals and adaptation.* *(p.5) in (Simon 1996)*

Artefacts denote a whole, contrary to components and parts. Objects and things also denote a whole, but the former could apply to non-physical things as in software engineering, and the latter incorporates both natural and artificial things. Hence, we call this level artefact.

### 4.6.1 LITERATURE ON ARTEFACT STRUCTURING

The obvious place to start the review on artefacts is Mereology or the theory of part-whole relations. This is a basic discipline for physical structuring. It is very theoretical, as it is thought out as a general theory for all kinds of part-whole structures. The discipline uses first order logic to formulate axioms that can be used to describe any kind of part-whole relation. A historical summary of the work done in the twentieth century is presented in *Parts* by (Simons 1987) and with a little more practical approach in *Parts and places* by (Casati & Varzi 1999). Both works are theoretical and cannot be applied directly, but they serve as conceptual inputs for later application.

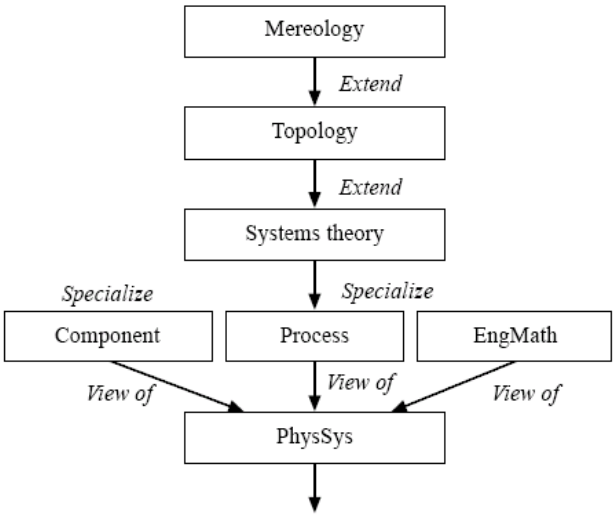



**Figure 72 - The ontological view with PhysSys (Borst, Akkermans, & Top 1997)**

This part-whole way of looking at artefacts has been put into use in the engineering ontology PHYSSYS by (Borst, Akkermans, & Top 1997). Their contribution is a standard vocabulary presented in an ontology based on other domains like Mereology, topology, system theory and mathematics so that reuse can be maximized when structuring artefact modelling. Their worldview is presented in Figure 72.

The use of part-whole relations in ontology is not the only way to utilize this theoretical work. A more practical approach is the bill of material well known in most companies.

Bill of materials or BOMs are used in ERP systems to describe products and their assemblies/components. Most companies use some sort of BOM in their structuring but they can take on very different appearances and are usually very individualized. Some work on making BOMs in a general form is presented in *General BOM* by (Hegge & Wortmann 1991), where the main aim is to achieve repeatable BOM structures. An example of generic BOM is shown in Figure 73; it relies on domain-dependant decomposition.

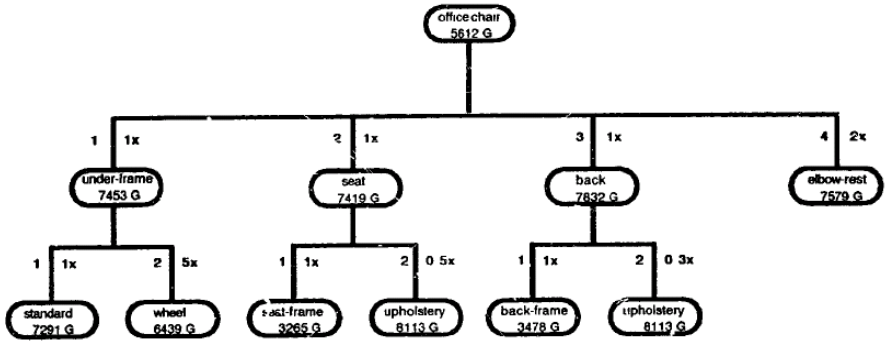



**Figure 73 - Generic bill of material of an office chair (Hegge & Wortmann 1991)**

The richest literature on physical structuring is most likely the modularity literature. Thinking about modularity is old, but one of the first formulations in the literature is by (Starr 1965), where he uses the term module in addressing the need for more product variety and hence more flexible structures. Some work was done in the 1970s and 1980s, but first in the 1990s, when the manufacturing systems became mature enough to support such concepts, did modularity gain the necessary attention in the research community. The definitions of (Ulrich & Tung 1991), with different kinds of modularity and later with focus on relations (Ulrich 1995), supply some of the building blocks for later use. For great reviews on modularity see: *Three faces of modularity* by (Fixson 2003); the definitions and benefits of modularity (Gershenson et al. 2003), where the authors support Fixson's findings on lack of "operationalizing" of modularity research and common definitions; the platform view by (Tollenaere & Jose 2005); and finally, the generality view of (Salvador 2007), where the author tries to make a single generic definition of the subject. The above-mentioned literature is practical in its approach, and the focus is on operations.

An excellent theoretical approach is offered by (Schilling 2000), where the author introduces a model of "Modular systems" with factors like: heterogeneity of inputs, synergistic specificity, urgency and heterogeneity of demands, as shown in Figure 74.

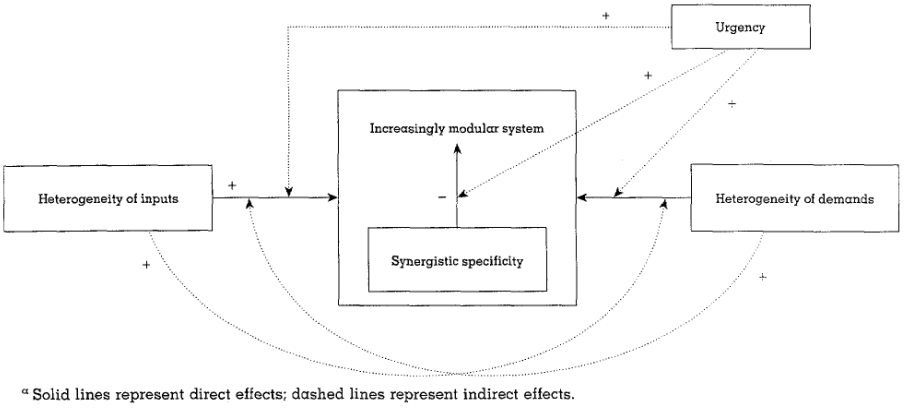



**Figure 74 - Modular systems (Schilling 2000)**

As this is not a work on modularity, this domain is viewed from the perspective of product models and how to construct such models. The work related to taxonomies (Bi & Zhang 2001), complex systems (Ethiraj & Levinthal 2004), life-cycle engineering (Gu & Sosale 1999), products and systems (Huang & Kusiak 1998), part reuse (Kimura et al. 2001) and modular interdependency in complex dynamical system (Watson & Pollack 2005) can provide inspiration for ways of modelling.

Many methods that aim at achieving modularity are clumsy at best; especially index methods are beyond researchers. The best by far are the *modular function deployment* of (Erixon 1998) and the process suggested by (Ulrich & Eppinger 2004).

Modular engineering (Miller 2001) offers a look at the distinction between building block and module. Module has to have a certain amount of functionality, while building blocks do not; thus, traditional Lego's © are building blocks and not modules. A twist on modularity is the products system view, where each product can be seen as a module; this is explored by (Langlois & Robertson 1992). The product system view is particularly relevant in this thesis, as the problem at the case company involves exactly this. This view is closely tied to modularity and its philosophy. A systemic way of structuring artefacts is presented in Figure 75.

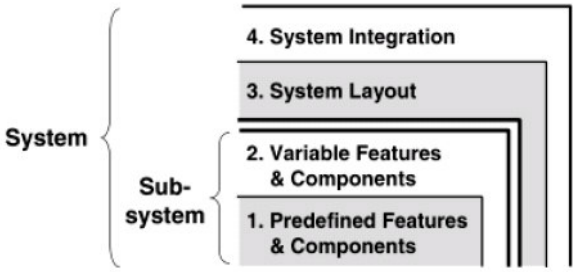



**Figure 75 - The hierarchic layers of the product architecture (Hofer & Halman 2004)**

Here, the main issue becomes managing interfaces so that the sub-systems can be combined into systems.

The part-whole decomposition of artefact structuring is the most common way to model a physical product. All the four methods mentioned above – the ontology, generic BOM, modularity and systems – rely on part-whole decompositions. For a mereological approach to artefact structuring, see both the sections on relations (p.86) and decomposition (p.93).

Classification can also be used as a main concept but only at family level, as seen in Figure 76. Classification fails at single product level, as the product cannot be classified into pieces. More on classification in the section on decomposition (p.93).

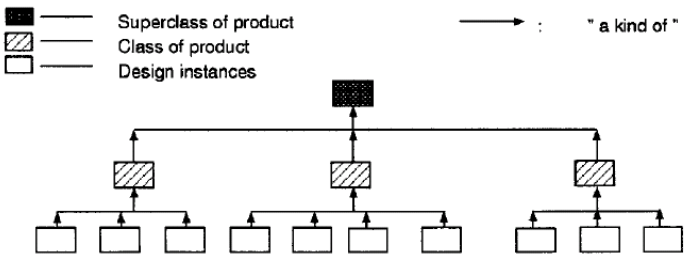



**Figure 76 - Product family classification tree (PFCT) (O'Donnell et al. 1996)**

The decomposition section deals with artefact structuring in models and states that all single product methods are driven by part-whole relations, and that classification are only usable on family level.

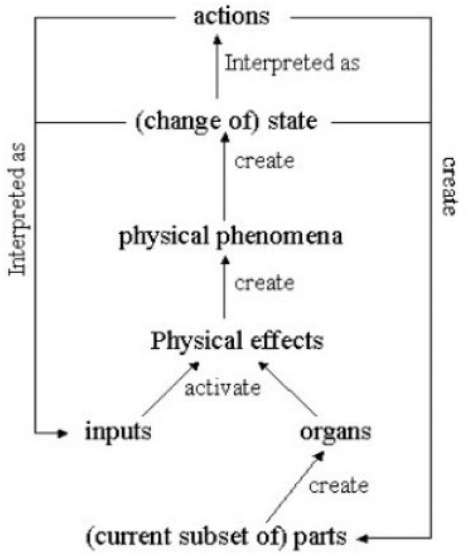



**Figure 77 - The SAPPhIRE model of causality**
**(Chakrabarti, Sarkar, Leelavathamma, & Nataraju 2005)**

A process to develop a suitable artefact structuring could be the sapphire model on causality as it takes into account many different aspects that we need to consider such as organs or modules, behaviour and states. The model is shown in Figure 77.

The seven elements in the models are:

1. *Parts*: A set of physical components and interfaces constituting the system and its environment of interaction.
2. *State*: The attributes and values of attributes that define the properties of a given system at a given instant of time during its operation.
3. *Organ*: The structural context necessary for a physical effect to be activated.
4. *Physical effect*: The laws of nature governing change.
5. *Input*: The energy, information, or material requirements for a physical effect to be activated; interpretation of energy material parameters of a change of state in the context of an organ.
6. *Physical phenomenon*: A set of potential changes associated with a given physical effect for a given organ and inputs.
7. *Action*: An abstract description or high-level interpretation of a change of state, a changed state, or creation of an input.

The relationships between these constructs are as follows: Parts are necessary for creating organs. Organs and inputs are necessary for activation of physical effects. Activation of physical effects is necessary for creating physical phenomena and changes of state. Changes of state are interpreted as actions or inputs, and create or activate parts. Essentially, there are three relationships: activation, creation, and interpretation.

Keeping this in mind, constructing a relevant artefact structure should be more guided. Note that this model is really inspired by the work of (Andreasen 1991) and (Hubka & Eder 1987). The different aspects of artefacts modelling are summarized in Table 16.

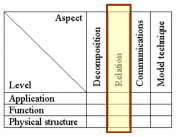
**Table 16 - Selected literature on physical structure**

| Author \ Levels & Aspects | Decomposition | Relation | Communications | Model technique |
|---|---|---|---|---|
| (Simons 1987) | ◐ | ● | | |
| (Hegge & Wortmann 1991) | ◐ | ◐ | | ◔ |
| (Krause et al. 1993) | ◐ | ◔ | | ◐ |
| (Ulrich 1995) | ● | ● | ◔ | ◔ |
| (Gerstl & Pribbenow 1996) | ● | ● | | ◔ |
| (Smith 1996) | ◐ | ◐ | | ◐ |
| (Varzi 1996) | ● | ◔ | | ◐ |
| (Artale et al. 1996) | ● | ● | | ● |
| (O'Donnell, MacCallum, Hogg, & Yu 1996) | ◔ | ◔ | | ● |
| (Tichem & Storm 1997) | ◐ | ◐ | ◔ | ◔ |
| (Gupta & Krishnan 1998) | | ● | | |
| (Juengst & Heinrich 1998) | ◔ | ◔ | ◔ | ◔ |
| (Newcomb et al. 1998) | ● | ● | | ● |
| (Erixon et al. 1996) | ◐ | ◐ | | ● |
| (Mortensen & Hansen 1999) | ● | ◔ | | ◐ |
| (Jiao & Tseng 2000) | ● | ◐ | ◐ | ◐ |
| (Mannisto et al. 2001) | ● | ◐ | | ◐ |
| (Sanchez & Collins 2001) | | ◐ | ● | |
| (Hansen & Rutahuhta 2001) | ◐ | | | ◐ |
| (Gershenson, Prasad, & Zhang 2003) | ◐ | | ◐ | |
| (Hofer & Halman 2004) | ◐ | ◐ | ◔ | |
| (Chakrabarti, Sarkar, Leelavathamma, & Nataraju 2005) | ◐ | ● | | ◐ |
| (Tollenaere & Jose 2005) | ◐ | ◔ | ◔ | ◐ |

The literature presented in Table 16 on artefact structure is very related to decomposition and relations. Most authors deal with all aspects, though communication is often left out. This is understandable as it is not necessary in a single artefact view. Key literature on product structuring is to be found in a single article *A conceptual theory of part-whole relations and its applications* by (Gerstl & Pribbenow 1996). It captures the essential core of making part-whole structures.

Let us now move on to two very related topics, namely relationships and decompositions. This section and the following two could have been merged into one, but the author feels it beneficial to look at these matters separately. Next section is on relation and the next after that on special kinds of relations, decompositions.

# 4.7 RELATIONS




This section deviates from the normal theoretical walkthrough and lists some of the details, because the author feels it necessary for the later stages and for understanding the general solution concept. Even though this section is named Relations, there are several synonyms that are fully equivalent, such as: dependencies, connections, association, link, interfaces, coupling, causality, rules and connection structure. We do not go into the nuances between the terms but focus on the fact that they all mean that two (or more) elements are related in some way. To start the discussion let us look at relations and how they can be defined.

## 4.7.1 WHAT IS RELATION?

Even though relations are one of the world's most important concepts, it has not been covered in depth for practical use. Mathematics uses relations extensively, and one definition is:

> *In mathematics, a binary relation (or a dyadic or 2-place relation) is an arbitrary association of elements within a set or with elements of another set. Binary relations are used in many branches of mathematics to model concepts like "is greater than", "is equal to", and "divides" in arithmetic, "is congruent to" in geometry, "is adjacent to" in graph theory, and many more. .... A binary relation is a special case of a* k*-ary relation, that is, a set of* k*-tuples where the* $j^{th}$ *component of each* k*-tuple is taken from the* $j^{th}$ *domain* $X_j$ *of the relation.*
>
> *(from* www.wikipedia.org*, accessed 20.11.2007)*

We assume that most of the relations used in this work are binary relations. This is done for the sake of simplicity. Dealing with *k*-ary relations is not easy. Another thing, a single definition of relation is not chosen, at least not here. But we walk through literature that deals with relations in some form, and see if we can generate a consensus of what should be included in such a definition.

## 4.7.2 LITERATURE ON RELATIONS

Literature that deals with relations is quite focused. It mainly comes from Mereology or the study of part-whole relations, but also from ontology, engineering design, requirement engineering and modelling. These domains do not treat relations on the same abstraction level, and while most are very theoretical, some take a more practical approach. It is the intention in this section to create a frame that can be used later in this thesis to generate taxonomy of relations relevant in modelling artefacts on many abstraction levels.

To lay the foundation, let us look at the building blocks of relations so we can later apply them to the field of Mereology. In Table 17, basic relations are identified as in Gruber's *Frame Ontology* (Gruber 1993).

**Table 17 - Elemental relation properties (Gruber 1993)**

| Relation type | Description |
|---|---|
| Relation (?Rel) | Relation (?Rel) defines a relation ?Rel in the domain. The classes to which the relation applies are defined as the domain and the range of the relation, respectively. |
| Sub-relation-of (?Child-Rel ?Parent-Rel) | A relation ?Child-Rel is a sub-relation of the relation ?Parent-Rel if, viewed as sets, ?Child-Rel is a sub-set of ?Parent-Rel. In other words, every tuple of ?Child-Rel is also a tuple of ?Parent-Rel, that is if ?Child-Rel holds for some arguments arg_1, arg_2… arg_n, then ?Parent-Rel holds for the same arguments. Thus, a relation and its sub-relation must have the same arity, which could be undefined. |
| Reflexive-Relation (?Rel) | Relation ?Rel is reflexive if ?Rel(x,x) holds for all x in the domain and range of ?Rel. |
| Irreflexive-Relation (?Rel) | Relation ?Rel is irreflexive if ?Rel(x,x) never holds for all x in the domain and range of ?Rel. |
| Symmetric-Relation (?Rel) | Relation ?Rel is symmetric if ?Rel(x,y) implies ?Rel(y,x) for all x and y in the domain and range of ?Rel. |
| Antisymmetric-Relation (?Rel) | Relation ?Rel is antisymmetric if ?Rel(x,y) implies not ?Rel(y,x) when x!=y, for all x and y in the domain and range of ?Rel. |
| Asymmetric-Relation (?Rel) | Relation ?Rel is asymmetric if it is antisymmetric and irreflexive over its exact domain. The exact domain of ?Rel is the set elements of the ?Rel range through this relation; that is, the exact domain only keeps the domain elements that participate in the relation. |
| Transitive-Relation (?Rel) | Relation ?Rel is transitive if ?Rel(x,y) and ?Rel(y,z) imply ?Rel(x,z), for all x and z in the domain and range of ?Rel, respectively, and for all y in the domain and range of ?Rel. |
| Equivalence-Relation (?Rel) | Relation ?Rel is equivalence relation if it is reflexive, symmetric and transitive. |
| Partial-Order-Relation (?Rel) | Relation ?Rel is a partial-order relation if it is reflexive, antisymmetric and transitive. |
| Total-Order-Relation(?Rel) | Relation ?Rel is a total-order relation if it is a partial-order relation for which either ?Rel(x,y) or ?Rel(y,x) holds for every x or y in its exact domain. |

The **I**ntegrated **DEF**inition methods (IDEF) have resulted in six models, from IDEF0 to IDEF5. The last one deals with constructing ontology. This is in itself not so important, but what is important is that the authors spend a lot of effort dealing with relations. The IDEF5 comes up with a relation library (see Figure 78) and good discussion on the groups suggested.

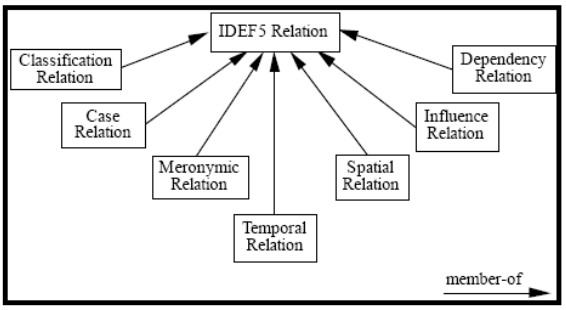



**Figure 78 - Overview of IDEF5 library relations (IDEF 1994)**

Based on the library relations from IDEF5 and using elemental relations, (Storga et al. 2005) went on to define groups of relations that relate more to the world and could be applied in engineering design. Their work, which is very theoretical and builds on Mortensen's GDMS (Genetic Design Model System)(Mortensen 1999), resulted in the following main terms extracted from GDMS and classified with the elemental relations presented in Table 18.

**Table 18 - Main relations from GDMS (Storga, Andreasen, & Marjanovic 2005)**

| | |
|---|---|
| CASE-ROLE RELATIONS: | The class of Relations relating the spatially distinguished parts of a Process. The relation is Anti-symmetric and Irreflexive by definition. Case-role includes, for example, the agent, patient or destination of a transformation. |
| are: | Instrument, Operand, Operator, Resource |
| Example: | (Operand ?Process ?Entity) means that ?Entity is a participant in ?Process that may be moved, changed, experienced etc. |
| CAUSAL RELATIONS: | The class of Relations that capture semantics of the fact that one concept has some effect or impact on another concept. The relation is Anti-symmetric, Irreflexive and Transitive. |
| are: | Aim, Causes, Changes, Consequence, Effect, Factor, Intention, Needs, Reason, Response, Result, Role, Stimulus |
| Example: | (Causes ?Process1 ?Process2) means that the ?Process1 brings about the ?Process2. |
| CLASSIFICATION RELATIONS: | The class of Relations that capture semantics of kinds and types. The relation is Anti-symmetric, Reflexive and Transitive. |
| are: | Is a, Instance of, Sub-kind of |
| Example: | (Subkind_of ?Machine ?Device) means that the ?Machine is sub-kind of Devices (that have a well-defined resource and result and that automatically convert the resource into the result). |
| GENERAL RELATIONS: | The class of Relations that capture semantics of very general predicates. |
| are: | Expresses, Knows, Inhibits, Possesses, Represents |
| Example: | (Represents ?Object ?Entity) means that ?Object in some way indicates, expresses, connotes, pictures, describes etc. ?Entity. |
| INTENTIONAL RELATIONS: | The class of Relations between an Agent and one or more Entities, where the Relation requires that the Agent has awareness of the Entity. The relation is Anti-symmetric and Irreflexive. |
| are: | Decision, Dislikes, Needs, Precondition, Wants |
| Example: | (Needs ?Agent ?Object) means that ?Object is physically required for the continued existence of ?Agent. |
| MERONYMIC RELATIONS: | The class of Relations that capture semantics of whole/part concept. The relation is Anti-symmetric, Irreflexive and Transitive. |
| are: | Component, Material of, Member, Mereological difference, Mereological product, Mereological sum, Part of, Proper part, Superficial parts |
| Example: | (Part ?EngineeringComponent ?Assembly) simply means that the Object ?EngineeringComponent is physical part of the Object ?Assembly. |
| PROBABILITY RELATIONS: | The class of Relations that permit assessment of the probability of an event or situation. The relation is Anti-symmetric and Irreflexive. |
| are: | Argument (Increase likelihood), Decrease likelihood |
| Example: | (Argument ?Statement1 ?Statement2) means that ?Statement2 is more likely to be true if ?Statement1 is true. |
| SPATIAL RELATIONS: | The class of Relations that capture semantics of the geometric, physical and other form of connections, contacts or interactions. The relation is Reflexive and Symmetric. |
| are: | Between, Connects, Connected, Contains, Interfaced, Located, Overlaps spatially |
| Example: | (Overlaps spatially ?Object1 ?Object2) means that the Objects ?Object1 and ?Object2 have some parts in common. |
| TEMPORAL RELATIONS: | The class of Relations that capture semantics of time-dependent relations. The relation is Anti-symmetric, Irreflexive and Transitive. |
| are: | Begin, Co-occur, End, Finishes, Follows, Future, Meets temporally, Overlaps temporally, Past, Proceeds, Relative time, Starts, Temporally between, Time relation, When |
| Example: | (Co-occur ?Process1 ?Process2) means that the Process ?Process1 occurs at the same time as, together with, or jointly with the Process ?Process2. |

More authors have tried to define relations that apply to the world and are based on the elemental properties of relations. One of the most relevant is the following definition of relations:

> *Connection relation*: *reflexive and symmetric*
> *Parthood relation: reflexive and transitive*
> *(p.366) in (Cohn & Varzi 2003)*

These two groups relate nicely to the suggested modelling method where *parthood* relations are used in the tree structure (PVM) and *connection* relations in the matrixes, both internal and mapping matrixes.

There are two major books in Mereology that are to be mentioned, *Parts* and *Parts and places*. The book *Parts* by (Simons 1987) is a historical summary of mereological work through the 20th century. It builds on propositional logic and is very theoretical. The most relevant part of the work is where Simon discusses "ontological dependencies", mentioning several. These dependencies only deal with artefacts (parts), and Simon states (in 1987) that no adequate functional dependency theory exists. This is also seems to be the case, still today.

The other book, *Parts and places* by (Casati & Varzi 1999) has stronger focus on spatial wholes. It is still theoretical and uses propositional logic. Casati & Varzi try to combine Mereology (concerned wit the concept of part) with topology (concerned with the concept of a connected whole) into mereotopology. Their work is a general part-whole theory, and their reason for it is given as Mereology's lack of dealing with relations. They talk about "connection structures" when they refer to relations and use topology to define them. This is supported in Varzi's later work; see (Cohn & Varzi 2003) mentioned earlier.

> *Among the various integrity relationships holding within a whole, the following distinction can be made:*
> *"Vertical" relationships.*
> - *Dependence relationships between the existence of the whole and the existence of (a certain number of) parts (and vice versa).*
> - *Dependence relationships between the properties of the whole and the properties of the parts (and vice versa).*
>
> *"Horizontal" relationships.*
> - *Constraints among parts which characterize the integrity of the whole.*
>
> *(p. 354) in (Artale, Franconi, Guarino, & Pazzi 1996)*

Most focus on vertical relationships and horizontals are largely ignored. A much deeper walk-through of this is found in (Simons 1987). What is important in relation to this thesis is that *horizontal relationship* cannot be ignored, as the system as a whole is largely defined by horizontal relations.

In contrast to the work mentioned until now, which is very theoretical and maybe not so applicable, several authors have tried to offer a more practical look at relation. Common use of relations is presented in (Sugumaran & Storey 2002), where the authors suggest that the three most commonly used relations are: *is-a*, *synonym* and

*related-to*. These are a little too generic for this thesis, but the authors also suggest a heuristic way of identifying relations, which is much more interesting.

Class taxonomies or relations between classes, as introduced in Frame Ontology (Gruber 1993), offers a way to construct part-whole relations, as shown in Table 19.

**Table 19 - Class taxonomies (Gruber 1993)**

| | |
|---|---|
| Subclass-of | (?Child-Class ?Parent-Class) |
| Superclass-of | (?Parent-Class ?Child-Class) |
| Disjoint-Decomposition | (?Class ?Class-Set) |
| Exhaustive-Decomposition | (?Class ?Class-Set) |
| Partition | (?Class ?Class-Set) |
| Instance-of | (?Individual ?Class) |

This kind of taxonomy would be quite relevant in software and therefore in the PVM structure. It does not deal with inherence, but in this thesis, this is not a problem. Another way of dealing with part-whole relations is the view offered in Mereology, here taken from (Artale, Franconi, Guarino, & Pazzi 1996) and presented as a list of different types (groupings) of relations in Table 20.

**Table 20 - Relation in mereology (Artale, Franconi, Guarino, & Pazzi 1996)**

| Six classes of: (Winston et al. 1987) | Evolved into | Four classes of: *(Iris et al. 1988)* |
|---|---|---|
| Component / Integral-object<br>Member / Collection<br>Portion / Mass<br>Stuff / Object<br>Feature / Activity<br>Place / Area | | Component-Whole<br>Segment-Whole<br>Member-Collection<br>Subset-Set |

This is not the only view offered in Mereology. One most excellent view offered on relation, both in Mereology but also in other domains, is the one presented by (Gerstl & Pribbenow 1996). Their practical and common sense has an operational aspect to it, so it can easily be applied when constructing relations. It applies equally to relations and decompositions, and a view is shown in Figure 79.

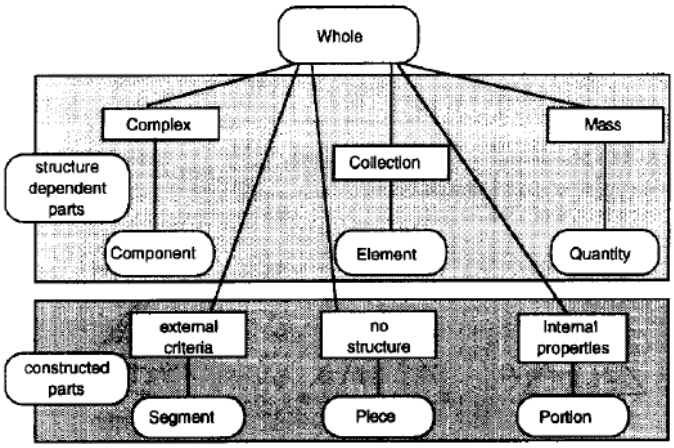



**Figure 79 - Classification of Part-Whole relations (Gerstl & Pribbenow 1996)**

The authors state that there are three main types of wholes: masses, collections and complexes, which are then represented with parts: quantities, members (elements), and components. These lead to two ways to isolate: intrinsic properties (portions) or external schemes (segments) (Gerstl & Pribbenow 1996). This is of course a further

evolution of the view offered by Iris et al. in Table 20, but the first two items are joined in complex/components.

Constructing ontology is very reliant on relations and their properties. In ontological engineering, it is important to pre-define such relation types, which has been done in several standards for ontologies, like *Frame ontology*, *RDF*, *OIL* and *OWL*. A fine summary of these and more can be seen in (Gomez-Perez, Fernandez-Lopez, & Corcho 2004). We discuss ontologies further in the chapter on communication on p.97.

However, dealing with the relation itself is not the only way to categorize relations. In their work on standardizing information retrieval for engineering design, Ramani & Li have constructed a set of relation groups that are closely related to physical things (Li & Ramani 2007). Their work differs because of the inclusion of parent and child in addition to the relation itself. The groups are common-sense like and directly applicable when describing artefacts in product modelling. The relations are shown in Table 21.

**Table 21 - Relationship groups in design information (Li & Ramani 2007)**

| Relationship | | | Definitions of the relationship |
|---|---|---|---|
| is-a | Child | Parent | Describes the generalization from a child concept to its parent concepts or the specification from a parent concept to its child concepts |
| has-part | DC | DC | Represents the part-whole between a DC and the other DC |
| has-function | DC | FC | Refers to the connection between a DC and one of its FCs |
| interface-with & Interact-with | DC | DC | Complement the has-function relationship when there is an 'object' in the function description of 'subject + verb [+ objects]'. Together, they represent the interactions between a DC and the other DC or EC |
| has-material | | EC | Describes the type of materials used in making the DC |
| has-process | DC | MC | Describes the type of manufacturing process used to make/fabricate the DC |
| use-material | DC | MFC | Describes the type of possible raw materials that certain manufacturing processes act on |
| has-property | MFC | MC | Each DC has several PCs characterizing its attributes, such as various physical attributes and geometry attributes |
| | DC/ MC | PC | Each MC may also have several PCs specifying its characteristics such as physical and mechanical attributes |
| has-measurement | | | Most of the PCs have one or several MUCs |
| has-value | PC | MUC | Each PC may have numerical VC or symbolic VC, while MUC only has numerical VC |
| has-feature | PC/ MUC | VC | Describes the significant shape features a device may have |
| has-standard | DC | SFC | Specifies the standard a DC/MC/MFC may comply with |

Legend for Table 21:

| | | | |
|---|---|---|---|
| DC: | device concept | FC: | function concept |
| MC: | material concept | MFC: | manufacturing process concept |
| SC: | standard concept | PC: | property concept |
| VC: | value type concept | SFC: | shape feature concept |
| EC: | environment concept | MUC: | measurement unit concept |

In relation to the requirements view on dependencies (Zhang, Mei, & Zhao 2006), the authors introduce four kinds of dependency between features: *refinement*, *constraint*, *influence* and *interaction*. Each of these can then be broken down into further divisions, e.g. refinement breaks down into *decomposition*, *characterization* and

*specialization*. This is not helpful in this thesis, because the definitions are vague, but what is interesting in their work is the focus on finding inconsistencies or anomalies in relations.

Yoo and co-authors (2004) offer another view on relations in requirements. They state that a generic, domain-independent relationship taxonomy exists, as shown in Figure 80.

| | |
|---|---|
| 1. Generalization/specialization | |
| Self | 2. Characteristic |
| | 3. Descriptive |
| | 4. Occurrence |
| Whole-part/composition | 5. Configuration/aggregation |
| | 6.Membership/grouping |
| 7. Classification/instantiation | |
| Comparison | 8. Equivalence |
| | 9. Similar/dissimilar |
| Association/dependency | 10. Ordering |
| | 11. Activity |
| | 12. Influence |
| | 13. Intentional |
| | 14. Socio-organizational |
| | 15. Temporal |
| | 16. Spatial |

**Figure 80 - General relationships in Relationship Analysis (Yoo et al. 2004)**

These relations were found through literature study conducted by the main author in her Ph.D. thesis. What is good in this work is again the heuristic presented to collect information on the relationships, what questions to ask etc. The last work to be mentioned here is one of the most used modelling languages, the UML. In this standard, relations are treated very generally, and it is left up to the modeller to find out how they are. The relations in UML are shown in Figure 81.

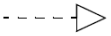
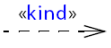
| *Relationship* | *Function* | *Notation* |
|---|---|---|
| association | A description of a connection among instances of classes |  |
| dependency | A relationship between two model elements |  |
| generalization | A relationship between a more specific and a more general description, used for inheritance and polymorphic type declarations |  |
| realization | Relationship between a specification and its implementation |  |
| usage | A situation in which one element requires another for its correct functioning |   |

**Figure 81 - UML relationship types (Rumbaugh et al. 2005)**

We deal a lot more with modelling techniques in a later chapter, so this will suffice for now. The selected literature in Table 22 shows that there are two groups of literature, the physical structure oriented group, coming mostly from Mereology and the function/feature/application oriented group coming from requirement engineering.

**Table 22 - Selected literature on relationships**

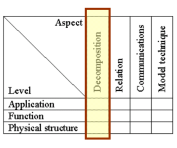
| Author \ Levels & Aspects | Application | Function | Artefact |
|---|---|---|---|
| (Simons 1987) | | ◔ | ● |
| (Gruber 1993) | | | ● |
| (Gerstl & Pribbenow 1996) | ◔ | ◔ | ● |
| (Artale, Franconi, Guarino, & Pazzi 1996) | ◔ | ◔ | ● |
| (Casati & Varzi 1999) | | | ● |
| (Cohn & Varzi 2003) | | | ● |
| (Gomez-Perez, Fernandez-Lopez, & Corcho 2004) | ● | ● | ● |
| (Yoo, Catanio, Paul, & Bieber 2004) | ● | | |
| (Bittner & Donnelly 2005) | | | ● |
| (Storga, Andreasen, & Marjanovic 2005) | ◔ | ◔ | ● |
| (Rumbaugh, Jacobson, & Booch 2005) | ◔ | ● | |
| (Zhang, Mei, & Zhao 2006) | ◐ | ● | |
| (Li & Ramani 2007) | ● | | |

As in earlier cases, no single work captures all aspects considered necessary here for understanding relations and for the proper taxonomy of them. The two views offer great insight and should be combined to use in the suggested solution.

For a single article on the subject, the comprehensible work, *A conceptual theory of part-whole relations and its applications* by (Gerstl & Pribbenow 1996), gives excellent insight into Mereology without being too theoretical. *Ontological Engineering* by (Gomez-Perez, Fernandez-Lopez, & Corcho 2004) deals with relations in depth and should be considered a good source.

A special type of relation is the decomposition of artefacts and concepts into smaller segments. All that has been said about relation also applies to decompositions, but this author feels it necessary to deal with this subject separately due to how the suggested solution is constructed.

## 4.8 DECOMPOSITIONS




Decompositions are only a special kind of relation. This ties the discussion of decompositions neatly to the relationship walkthrough. Remember, that some of the elemental relations dealt with decompositions, also called taxonomies. This chapter will focus more on the "process" part of decompositions and leave most of the "relational" part to the *Relations* chapter on p.86. Implementation of decomposition presumes that things are composed. Let us look at what decomposition is and is not.

### 4.8.1 WHAT IS DECOMPOSITION?

To decompose or break things down into smaller pieces is something most can do intuitively. Decomposition can be defined thus:

> *In a composition hierarchy, the set is decomposed repeatedly, each part being split into its child components.*
> *(p.75) in (Stevens, Brook, Jackson, & Arnold 1998)*

Partition is synonymous with decomposition. Ontology can be used as a guideline when decomposing. In contrast, classification is generic to specialized hierarchy. And taxonomy is a formalization of classification, a sort of guideline. In some cases, classification can be used instead of decomposition. The thing to remember here is that classification is not the same as decomposition.

### 4.8.2 LITERATURE ON DECOMPOSITION

Decomposition does not seem to be a 'sexy' area, as only a limited amount of literature exists on the subject. Most seem to discard decompositions as trivial and spend no time explaining how to decompose. This author feels that this is not good, if we are to achieve any kind of "standardization" in making models. In this section, we look at literature that tries to address the issue of decomposing.

First on our list is part-whole literature or Mereology. A great article by (Gerstl & Pribbenow 1996) presents two decomposition suggestions:

> *Partitions based on the compositional structure of the whole and Partitions of the whole which are arbitrary, or driven by internal features or external criteria.*
> *(p.306) in (Gerstl & Pribbenow 1996)*

These should be seen in conjunction with Figure 79. This common-sense approach to decomposition is excellent and points to two driving forces, *internal features* and *external criteria*.

The three views on decomposition offered by (Kusiak & Larson 1995) are: *product decomposition*, *problem decomposition* and *process decomposition*. The first can be driven by either internal or external criteria, while the latter two are driven by external. A problem decomposition with configuration in mind is presented by (Magro & Torasso 2003), where they define two kinds of decompositions: *Requirements-based decomposition* and *Constraints-splitting decomposition*. The former is problem description while the latter product view. Constraints-splitting decomposition is based on knowledge that constraints usually link together the components and the sub-components, and that these can be *bound* in such a way that a choice made in order to satisfy one of them may restrict the choices available for satisfying another one.

Different types of problem decompositions are offered as decomposition strategies in Figure 82.

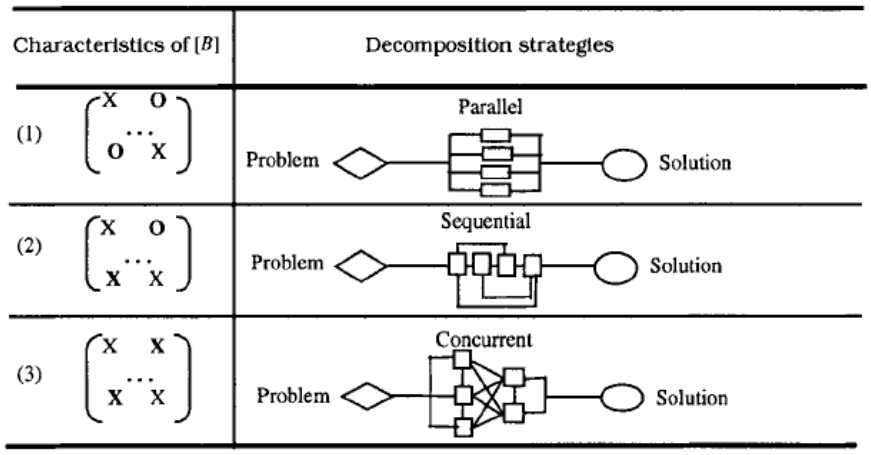



**Figure 82 - Decomposition methodologies (Bi & Zhang 2001)**

Figure 82 is based on Suh and his axiomatic design. Suh has fine logic for keeping design "clean" with focus on dependencies between FR (functional requirement), DP (design parameters) and PV (process variable), as shown in Figure 50 (Suh 1998). There are two facts that should be recognized by all designers:

1. *FRs and DPs have hierarchies, and they can be decomposed.*
2. *FRs at the ith level cannot be decomposed into the next level of the FR hierarchy without first going over to the physical domain and developing a solution that satisfies the ith level FRs with all corresponding DPs.*

*(p.36.) in (Suh 1990)*

This is a very important aspect, supported by other authors like Andreasen in his domain theory (Andreasen 1991), Simon in the architecture of complexity (Simon 1996), and Pahl & Beitz in their design process (Pahl, Beitz, Feldhusen, & Grote 2007). Functional decompositions, like those from (Pahl, Beitz, Feldhusen, & Grote 2007), the German engineering association (VDI 1986), or (Stone & Wood 2000), all rely on definition of elemental functions, which are combined to form more complex ones. This is very good in itself, but it can be quite complex to construct these hierarchies from the bottom up, as the details in the elemental functions is very high and far from the visible effect required.

Another approach uses elements in the integration analysis methodology suggested by (Pimmler & Eppinger 1994).

> *The first step in the analysis requires specification of the overall product concept in terms of functional and/or physical elements. This step is usually straightforward. Indeed, complex problems are quite commonly broken down into simpler sub-problems. The challenge, however, is in determining how finely the elements should be divided. (p.345) in (Pimmler & Eppinger 1994)*

To deal with the problem of elemental units, Pimmler & Eppinger suggest that the system should be broken down one step too far and then the elements combined into chunks, as shown in Figure 83.

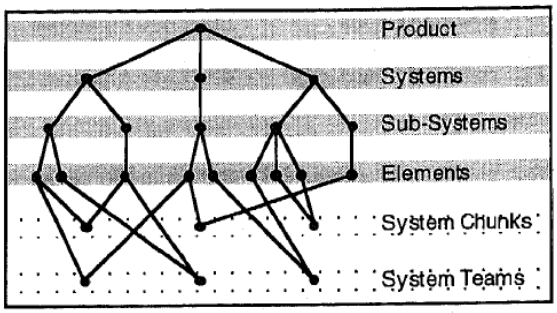



**Figure 83 - Integration methodology (Pimmler & Eppinger 1994)**

Although the matrixes used in (Browning 2001) offer no new view on decomposition, his work is not without merit. It combines several views with the use of DSM and offers some "straight-forward" points on how to construct a project from product, process and team perspective. An organizational view on task partitioning given by (Hippel 1990) is useful for team making but not so helpful in decomposition strategies for a functional approach. A quick summary of literature focusing on decompositions is shown in Table 23.

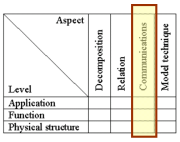
**Table 23 – Selected literature on decomposition**

| Levels & Aspects / Author | Application | Function | Artefact |
|---|---|---|---|
| (Shupe et al. 1987) | ◐ | ◐ | ◐ |
| (Hippel 1990) | ● | | |
| (Pimmler & Eppinger 1994) | ● | ● | |
| (Kusiak & Larson 1995) | ◐ | ● | ● |
| (Gerstl & Pribbenow 1996) | | | ● |
| (Kirschman & Fadel 1998) | | ● | |
| (Suh 1998) and (Suh 1990) | ● | ● | ● |
| (Stevens, Brook, Jackson, & Arnold 1998) | | ● | |
| (Mortensen & Hansen 1999) | ◔ | ◔ | ◔ |
| (Browning 2001) | | | ● |
| (Bi & Zhang 2001) | ● | | |
| (Stone & Wood 2000) | | ● | |
| (Magro & Torasso 2003) | ● | | ● |
| (Pahl, Beitz, Feldhusen, & Grote 2007) | ● | ● | ● |

For a single article on decomposition, read *Decomposition and representation methods in mechanical design* by (Kusiak & Larson 1995) or *Integration analysis of product decomposition* by (Pimmler & Eppinger 1994).

The next aspect to look at is communication, then followed by modelling techniques.

# 4.9 COMMUNICATION




The act of communicating is fundamental to our society. As we keep making more and more artefacts that are ever more complex, and which are then combined to form complex systems, the importance of communication is not restricted to human communication. To solve our complex artefact systems, we now require (or at least wish) them to communicate internally. This section looks at the act of communicating in the perspective needed for this thesis, i.e. basic cybernetic communication in restricted domains with predefined scope. Communication in this section is considered synonymous with both knowledge transfer and the act of speech, although in some domains this may be regarded as a little optimistic.

Most research into communication comes from two main areas, cybernetics (again in its broadest sense, which includes psychology, linguistics, and behavioural science). Let us start by finding out what communication is.

## 4.9.1 WHAT IS COMMUNICATION?

The mathematical theory of communication by (Shannon & Weaver 1949), and their three-fold view, with *technical problem*, *semantic problem* and *effectiveness problem,* is still today one of the most comprehensive analyses of communication and can be illustrated as shown in Figure 84:

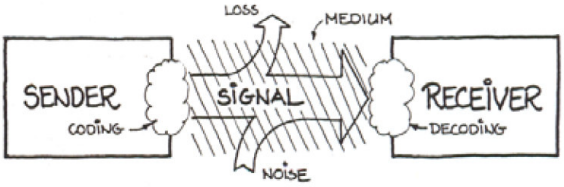



**Figure 84 - Communication theory (Shannon & Weaver 1949) quoted in (Buur & Andreasen 1989)**

The technical aspect is the coding, decoding, signal and medium; the semantic is the message in the signal; and the effectiveness is the noise and loss. A very similar model is offered by R. Jakobson in *Closing statement: linguistics and poetics* from 1960 and is shown in Figure 85.

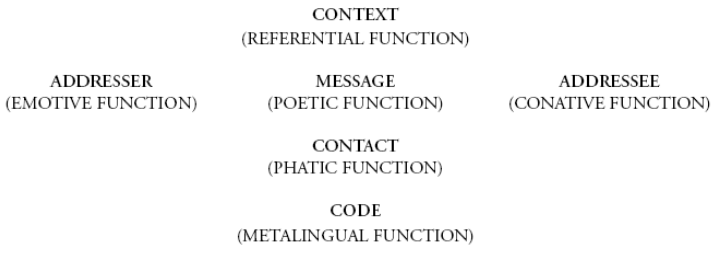



**Figure 85 – Jakobson's communication model, quoted in (Thoriacius 2002)**

In Jakobson's model, the environment is made explicit in *context,* while the communication theory only implicitly treats the environment through other aspects (coding, decoding, medium, noise and loss). Both these models separate the communication into elements that can be analysed. A similar effort was made by Lind in his MFM (multilevel flow model) method (Lind 1990). His view of communication draws four levels or layers that can be treated separately, as shown in Figure 86.

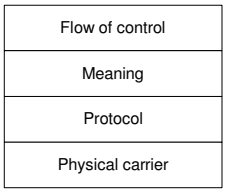



**Figure 86 - Communication layers (Lind 1990).**

Here, the technical and effectiveness problem connect to three layers (*physical carrier*, *protocol* and *flow of control*), and the semantic problem has its own layer in *meaning*. This model actually highlights the importance of the semantic problem. A merger of all three of these models used in the thesis as concept for understanding communication. Let us now look at some literature and the main concepts presented on communication there.

### 4.9.2 LITERATURE ON COMMUNICATION

This thesis is not about physical carriers or their design. It is about knowledge engineering. Therefore, we do not deal so much with the technical or effectiveness problems but focus on the semantic problem, i.e. the meaning of communication. This helps to focus the literature search and review. Let us start by looking at how humans communicate meaning and then move towards how to construct artefact systems that can achieve some sort of communication.

Communicating is something we humans have done throughout the millennia. Although a historic look at communication from the dawn of time serves no purpose, we can start our review in the 1960s with the seminal work of John R. Searle. His work is rooted in semiotics or the study of signs. One current in semiotic research is based on Mead's analysis of the act (Mead 1938). Here, a distinction is made between three dimensions of significations of signs, namely a *designative*, an *appraisive* and a *prescriptive* dimension. These three dimensions of signification are derived from Mead's decomposition of an act into three phases, as seen in Figure 87.

| Action requirements | Dimension of signification | Interpretant | Signification |
|---|---|---|---|
| Obtaining information | Designative | Sense organs | Stimulus properties of object |
| Selection of objects for preferential behavior | Appraisive | Object preferences Of the agent | Reinforcing properties of object |
| Action on object by specific behavior | Prescriptive | Behavior preferences of the agent | Act as instrumental |

**Figure 87 - Three dimensions of signification (Mead 1938)**

The work of Searle is based on Mead's work and can be considered the founding stone of how communication semantics is perceived in modern technologies. In his book, *Speech Acts* (Searle 1969), the author identifies different types of speech acts, as shown in Table 24.

These types lay the foundation for a later incarnation, where the human is replaced by a machine (or software), and the speech act is boiled down to even simpler types. We now "jump" over two decades in the evolution of speech act, since it is not important to this thesis, and take up the subject again in the 1990s, when work on intelligent agents became a fad in the artificial intelligence community. Here, due to the need to allow

agents of non-human nature to communicate autonomously, a "standardization" of the meaning of communication is required.

**Table 24 - Speech act types (Searle 1969)**

| | |
|---|---|
| *representatives:* | informs, e.g. 'It is raining.' |
| *directives:* | attempts to get the hearer to do something, e.g. 'please make the tea.' |
| *commisives:* | commits the speaker to doing something, e.g. 'I promise to… ' |
| *expressives:* | speaker expresses a mental state, e.g. 'Thank you!' |
| *declarations:* | e.g. declaring war or christening |

This work of defining the "building blocks" of communicational meaning and the syntax for their use went through several attempts before arriving at the FIPA (Foundation for Intelligent Physical Agents) standard in 2002. One of the best-known intermediary steps in ACL (agent communication languages) development is the framework developed by ARPA (Advanced Research Projects Agency of the United States Department of Defense), the KIF (knowledge interchange format)(Genesereth & Fikes 1992) for meaning and the KQML (knowledge query and manipulation language) for syntax. For a historical summary of the evolution of agent languages, see (Wooldridge & Jennings 1995). An excellent summary of how-to-construct agents and systems of agents is the bible, *Artificial Intelligence: A modern approach,* by (Russell & Norvig 2003), where all aspects of agents are explained. It is a summary work not an original contribution, but still a very good book.

In general, a speech act can be seen to have two components:

*performative verb:* (e.g. request, inform, promise etc.)
*propositional content:* (e.g. "The door is closed.")

In the KIF/KQML framework, the former deals with content and latter with performatives. In this thesis, the communication needed is defined and summarized nicely in the newly made FIPA standard. A recapitulation of the FIPA is in order.

| Parameter | Category of Parameters |
|---|---|
| performative | Type of communicative acts |
| sender | Participant in communication |
| receiver | Participant in communication |
| reply-to | Participant in communication |
| content | Content of message |
| language | Description of Content |
| encoding | Description of Content |
| ontology | Description of Content |
| protocol | Control of conversation |
| conversation-id | Control of conversation |
| reply-with | Control of conversation |
| in-reply-to | Control of conversation |
| reply-by | Control of conversation |

**Figure 88 - ACL message parameters (FIPA 2002b)**

Basic structure of the FIPA framework is quite similar to KQML:

- *performative:* 22 performative in FIPA; see (FIPA 2002c)
- *housekeeping:* e.g. sender etc.; see (FIPA 2002b)
- *content:* the actual content of the message; FIPA uses Semantic Language (SL) (FIPA 2002e), although they also experimented with KIF and RDF

More standards are available from FIPA, like architecture (FIPA 2002a), construction of ontology service (FIPA 2001), and device ontology (FIPA 2002d). An overview of the message structure suggested by FIPA is shown in Figure 88.

The two basic performatives in FIPA are *Inform* and *Request*. All others are macro definitions, defined in terms of these. The meaning of inform and request is defined in two parts:

- *pre-condition:* what must be true in order for the speech act to succeed
- *rational effect:* what the sender of the message hopes to bring about

The *inform* performative: the content is a *statement* and the pre-condition is that the sender:

- holds that the content is true
- intends that the recipient believe the content
- does not already believe that the recipient is aware of whether content is true or not

The *request* performative: the content is an *action* and the pre-condition is that the sender:

- intends action content to be performed
- believes recipient is capable of performing this action
- does not believe that recipient already intends to perform the action

With these two building blocks for communication, we can look at how to go about constructing a coherent framework for communication. In addition to these two concepts, there are two more concepts that are equally important: the control of communication and information storage. The former is dealt with in FIPA (see *participants in communications* and *control in conversation* in Figure 88) and is a higher abstraction concept outside the scope of FIPA. But it is a very important aspect of constructing a practical application of distributed AI. Let us draw on knowledge engineering to frame the second concept mentioned. Schreiber and his co-authors suggest a four-sided view based on who does the informing and the requesting, laced with the control of conversation; their view is presented in Figure 89.

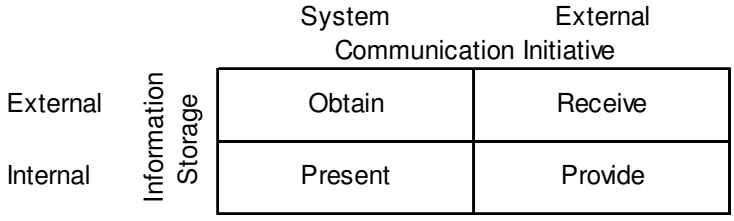



**Figure 89 - Communication initiative / storage matrix**
**(Schreiber, Akkermans, Anjewierden, Hoog, Shadbolt, van de Velde, & Wielinga 2000)**

In this view, two agents, system and external, have to share information. Depending on who can start the conversation, a mapping of inform/request would give the following sequences (system:inform should be read as system- does-inform):

*Present* system:Inform
*:*
*Obtain:* system:Request => external:Inform
*Provide* external:Request => system:Inform
*:*
*Receive* external:Inform
*:*

One of ontology's main application areas is "Tell & Ask" or communication. Note that tell & ask is just a rephrasing of inform/request. In their work on task ontology, (Mizoguchi, Vanwelkenhuysen, & Ikeda 1995b) suggest using ontology to define communications. As shown in Figure 88, FIPA has incorporated this possibility, although named *description* in the standard. Ontology helps restrict or narrow the possibility given in the content of messages. We have no intention of solving all the worlds' communications problems here; our need is domain specific and can be restricted to this. There are some excellent guidelines for constructing an ontology, e.g. (Noy & McGuinness 2001) and (Ahmed et al. 2007). In *Ontology Development 101* by (Noy & McGuinness 2001), the authors define a seven-step approach:

1. Determine the domain and scope of the ontology
2. Consider reusing existing ontologies
3. Enumerate important terms in the ontology
4. Define the classes and the class hierarchy
5. Define the properties of classes - slots
6. Define the facets of the slots
7. Create instances

The other approach focuses on engineering design and is built up more theoretically. It also "connects" ontology and taxonomy, and some might have a problem with that. But we do believe the method suggested to be sound and also to hold for construction of ontology. The method is shown in Figure 90.

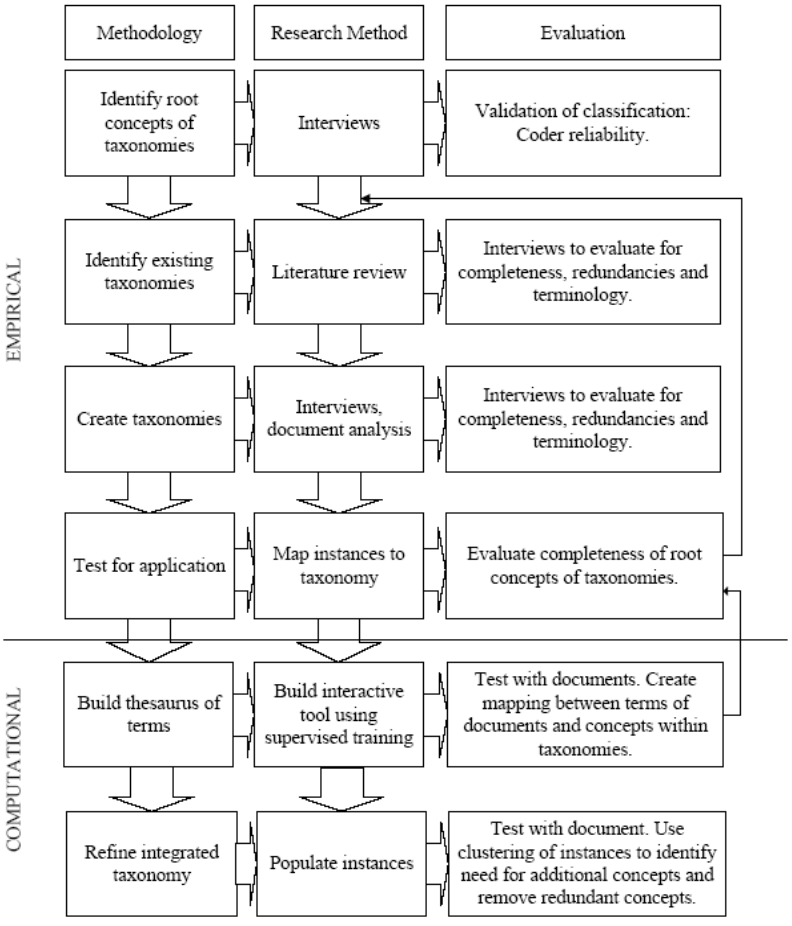



**Figure 90 - Methodology for development of an integrated taxonomy (Ahmed, Kim, & Wallace 2007)**

These methods are about organization of information after it has been figured out what is needed. To help us structure our thoughts with regard to the formulation of the information needed, the sense-making theory (Dervin 1998), including how to better structure information exchange, provides an excellent tool. Its core is "asking the right questions" when communicating. Think about the following: You go to the library and ask for Shannons & Weavers article on communication. The librarian gives you what you ask for, but you later realize that it was not exactly what you where looking for. If, in the beginning, you could explain to the librarian why you were there, like: "I am working on how to communicate knowledge and need literature related to communication", the librarian would have had a chance to "interpret" or put into "context" what you wanted and find something relevant. When communicating information, the receiver has the possibility to "interpret" or "contextualize" it to multiple sets of data. Thus, data, like communication, can work as a restriction on the performance of the whole system; but it requires that the receiver can interpret the information given.

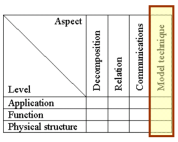
Let us now summarize some literature on communication (Table 25) with regard to the aspects presented at the beginning of this section. Most literature in Table 25 deals with communication from an application viewpoint. Thls is not surprising, as that is the whole point of communication. Some try to incorporate other views in order to be more complete in their approach. For a single article/report on the topic, look at *Abstract architecture specification* (FIPA 2002a) standard.

**Table 25 - Literature on communication**

| Levels & Aspects / Author | Application | Function | Artefact |
|---|---|---|---|
| (Shannon & Weaver 1949) | ◔ | ◔ | ● |
| (Searle 1969) | ● | | |
| (Lind 1990) | ◔ | ● | ◔ |
| (Mizoguchi, Vanwelkenhuysen, & Ikeda 1995b) | ● | | |
| (Dervin 1998) | ● | | |
| (Hopgood 2000) | | ◔ | ● |
| (FIPA 2002c) | ● | ◔ | ◔ |
| (Salles et al. 2001) | ● | | |
| (Noy & McGuinness 2001) | ● | | |
| (Russell & Norvig 2003) | ● | ● | |
| (Gomez-Perez, Fernandez-Lopez, & Corcho 2004) | ● | ● | |
| (Ahmed, Kim, & Wallace 2007) | ● | ● | |

That concludes our look at communication and how to construct it in a formalized manner. The final aspect in this theory review is modelling techniques.

# 4.10 MODELLING TECHNIQUES




An abstraction of the world is a model of the world. There are many kinds of models, from rigid mathematical models (like Newton's law) to visualizations of phenomena. In this section, we wish to look at modelling techniques that help us better understand artefacts and hence lead to better design of artefacts. The world is complex, and we mere humans cannot possible understand it as a whole. We need to break it down into manageable parts, and then model these parts to a higher abstraction for enhanced understanding.

A large portion of our brain deals with our vision and processing three dimensions and the patterns within them. This knowledge should drive us to make our models visual and use our great capability to deal with the 3D world. That should work to our advantage. An excellent rationale for visualization of knowledge is given by (Wexler 2001). Here, the author presents the positive aspects of knowledge mapping and four categories within which benefits can be measured: *Economic*, *Structural*, *Organizational culture* and *Knowledge returns*. We are not going to try this here, but will just sleep soundly knowing that it can be done, and accept the fact that it is a wise decision to visualize knowledge.

A modelling environment should have five quality-related and production-related properties; it should:

1. *nurture the entire modeling life-cycle, not just part of it;*
2. *be hospitable to decision and policy makers, not just to MS/OR professionals;*
3. *facilitate ongoing evolution of the models and systems built within it;*
4. *enable all of its inhabitants to 'speak' the same paradigm-neutral language for model definition;*
5. *facilitate good management of key resources, namely data, models, solvers, and knowledge derived from these.*

*(p.34) in (Geoffrion 1989)*

With these points in mind, we can look at several modelling methods, their strengths and weaknesses, and hints on how they could be improved or combined.

## 4.10.1 LITERATURE ON MODELLING

There are at least four groups of modelling techniques: matrix-based, tree-like, multi-graphs and UML-like (a specialization of multi-graphs). Each of these has its merit; the matrixes are really good at visualizing binary relations; the trees provide a very intuitive way of structuring hierarchical data; and multi-graphs are usually good at showing complex relational structures. The main difference in trees and multi-graphs is that branches in trees only connect to stems, whereas elements in multi-graphs can connect to multiple elements, e.g. the capacity for showing complex relations is much greater in multi-graphs.

A summary of each group is in order, starting with matrixes. The design structure matrix (DSM) suggested by (Steward 1981) is a good place to start, even though it surely has some predecessors.

Matrix techniques in product modelling can be categorized into intra-domain, inter-domain, product-level and matrix methodologies (Malmqvist 2002), depending on the elements noted in the rows and columns regarding their abstraction and whether the rows and columns have the same or different elements. The categories are shown in Figure 91.

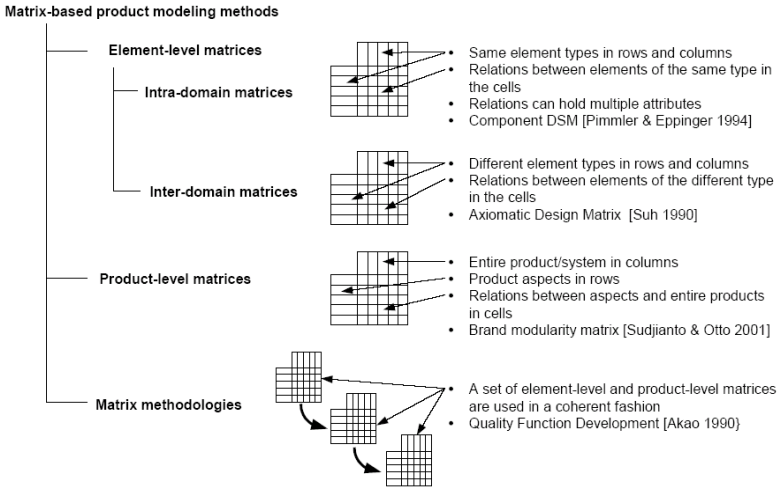



**Figure 91 - Overview of matrix-based product modelling method types (Malmqvist 2002)**

A similar usage of matrixes is presented in (Sage & Armstrong Jr 2000). Here, the authors introduce two kinds of matrixes, the Cross-interaction and Self-interaction matrixes. A self-interaction matrix is well known from QFD and is shown as half-a-matrix. It is the same as intra-domain matrixes but with half removed, as the relations are bidirectional. The cross-interaction matrix is an inter-domain matrix. Both matrixes are shown in Figure 92.

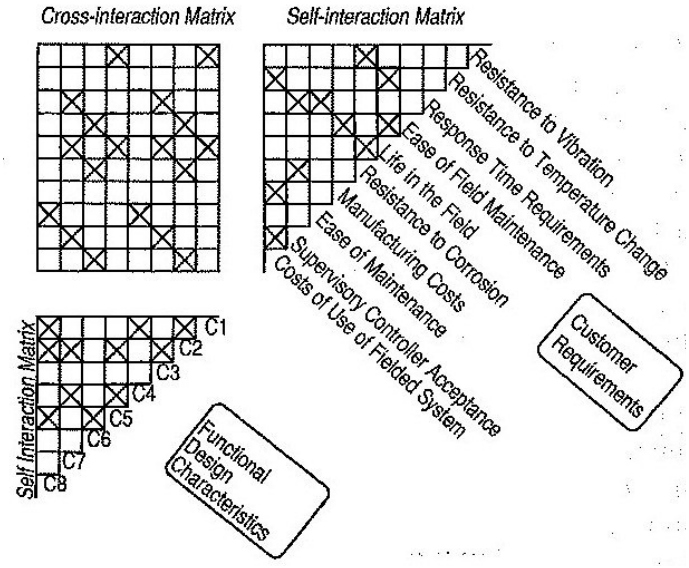



**Figure 92 – Cross- and Self-interaction matrixes (Sage & Armstrong Jr 2000)**

A tree model is a visualization of either decomposition or categorization. It can be a left-right or top-down tree but both would adhere to the same rule: branches only

connect to one stem. Some authors have tried to add more relations between branches, but there is a limit to how many can be added. An example of tree model with added relation is shown in Figure 93.

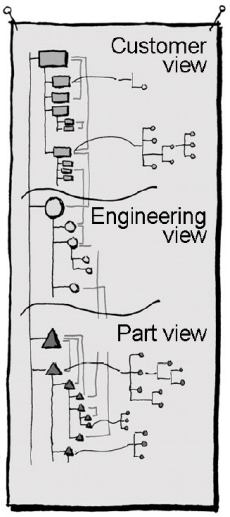



**Figure 93 - Product Variant Master (PVM) (Harlou 2006)**

Multi-graph methods remove the limitation of trees and allow multiple relations between elements. This can be realized in different ways. A typical multi-graph method is the graph in functional basis as seen in Figure 94.

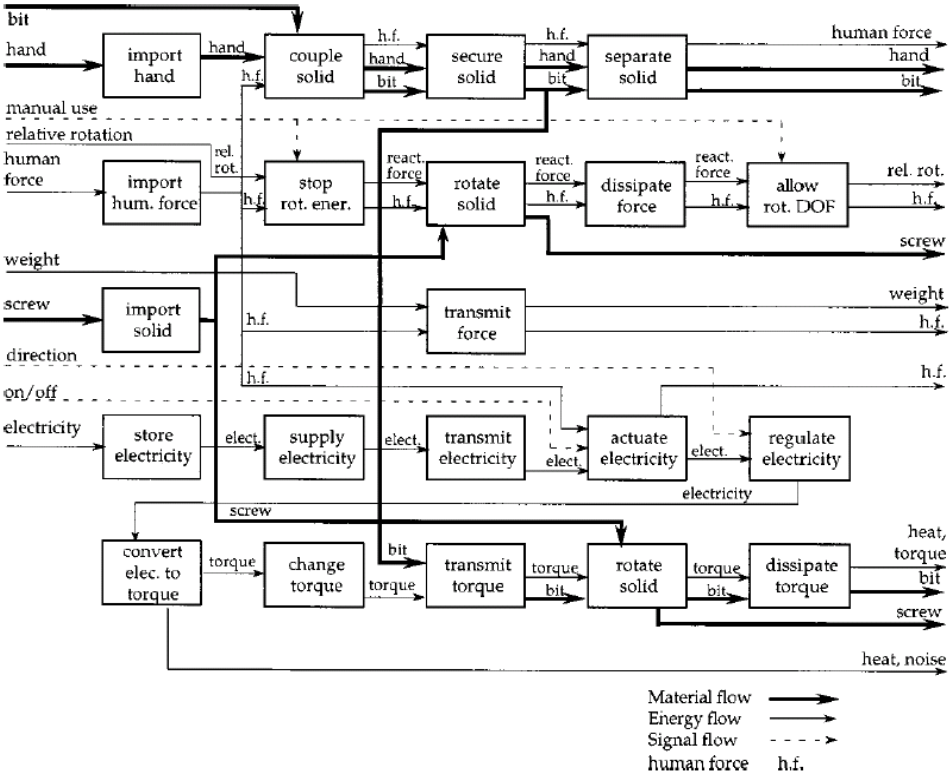



**Figure 94 - Functional model in Functional Basis (Stone & Wood 2000)**

The unified modelling language (UML) is a special multi-graph method with quite strict rules. It is aimed at software development but can be used in anything. There are several variants of UML to address different environments, and most of them work as extensions to the original. The class diagram in UML is a multi-graph, as shown in Figure 95.

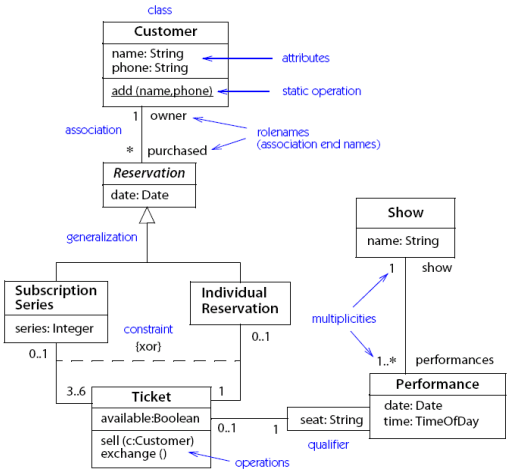



**Figure 95 - Class diagram in UML (Rumbaugh, Jacobson, & Booch 2005)**

System modelling language (SysML) is based on UML 2.1 (a part of UML called UML4SysML) with extensions that add specific things related to system engineering. The diagrams added can be seen in Figure 96.

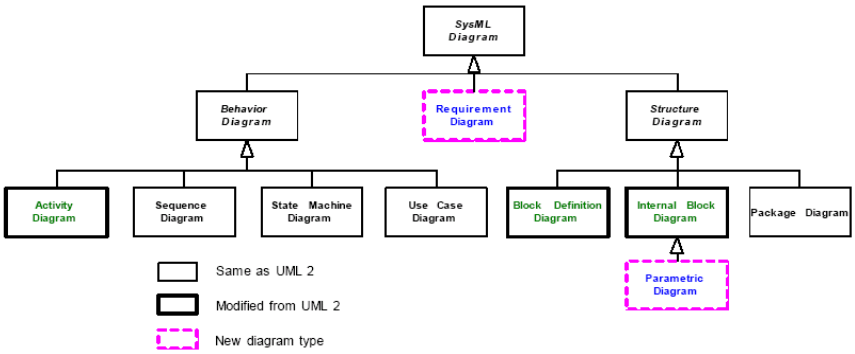



**Figure 96 - SysML diagram taxonomy (OMG 2007)**

The important addition in SysML is the *parametric diagram,* where parameters and their relations can be drawn, and what is input and what is not. An example of this diagram is shown in Figure 97.

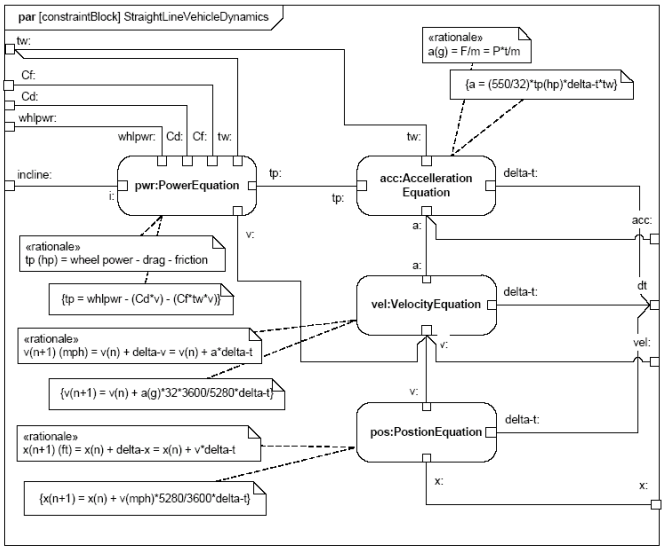



**Figure 97 - Example of Parametric diagram (Vehicle dynamics) (OMG 2007)**

This diagram offers an alternative to visualizing complex parameter setups.

Even though each of these modelling types has their merits, they all break down if the number of parameters and relations grows too large, i.e. the methods cannot show all parameters and relations in a single picture. Many might not consider this a problem, but it is one of the premises of this thesis that this is actually quite problematic when constructing electro-mechanical artefacts, especially when these are then grouped together to form a coherent whole or a system.

A comparison of different modelling methods and which abstraction levels they try to incorporate is shown in Table 26.

**Table 26 - Literature on modelling**

| | Technique & Author | Application | Function | Artefact |
|---|---|---|---|---|
| Matrixes | DSM (Steward 1981) | | | ● |
| | Design matrixes (Suh 1990) | ● | ● | ● |
| | QFD (Akao 1990) | ◐ | ◐ | ◐ |
| | MFD (Erixon 1998) | ◐ | ● | ● |
| | Structural Influence index (Wakefield 2005) | | | ● |
| | Matrix representation & mapping (Chen & Rao 2005) | ◐ | ◐ | ◐ |
| | DSM in configuration design (Helo 2006), | ◔ | ◐ | ◐ |
| | DMM (Danilovic & Browning 2007) | ● | ● | ● |
| | ESM (Bartolomei 2007) | ● | ● | ● |
| Trees | Function-means (Malmqvist 1997) | ◔ | ● | ◔ |
| | (Mortensen & Hansen 1999) | ◐ | ◐ | ◐ |
| | GTST (Modarres & Cheon 1999) | | ● | ● |
| | PFMP (Harlou 2006) | ◐ | ◐ | ◐ |
| UML var | UML (Rumbaugh, Jacobson, & Booch 2005) | ● | ● | ◐ |
| | UML for KBS (Felfernig et al. 2000) | ● | | ◐ |
| | MML (MOKA 2000) | ● | ● | ◔ |
| | SysML (OMG 2007) | ● | ● | ● |
| Multigraph | IDEF0 (IDEF 1993) | | ● | ◔ |
| | IDEF5 (IDEF 1994) | ● | | |
| | Functional Basis (Stone & Wood 2000) | ◔ | ● | ◔ |
| | Semantic nets (Hartley & Barnden 1997) | ● | | |
| | MFM (Lind 1990) | ◐ | ● | ● |
| | GFM (Soerensen 1999) | ● | ● | ◐ |
| | Requirements modelling (Evermann & Wand 2005) | | ● | |

The majority of methods presented in Table 26 deal with all abstraction levels, at least to some extent. They are largely divided by two factors: how stringent the visual representation rules are, and how well they deal with relations visually. Trees are actually quite fixed in their construct; we intuitively place elements in a visually pleasing pattern as if we were trying to mimic actual trees. Multi-graphs and especially

UML, even though they do have stringent rules for selection of forms and relations, do not have "placement" rules that guide us regarding how to make the diagrams visually pleasing. Therefore, such diagrams are often a 'mess'. To prove this point, just check out the diagrams used in both OMG standard manual and Rumbaugh's guide (Rumbaugh, Jacobson, & Booch 2005). No single reference covers all methods. A summary on matrix methods is found in (Malmqvist 2002). A typical multi-graph method is presented by (Stone & Wood 2000).

This concludes our walk-through of the different relevant theories. Many of these references have served as inspiration and have aided in generating the solution suggestion. Some are directly applicable, while most have aided in a conceptual way.

Again, the direction of this research is very much in line with the work done at MIT's MERS (Model-based Embedded and Robotic Systems) (Williams et al. 2003). Embedding knowledge into the system is the foundation here, but the work differs in focus: MERS is about showing actual implementations, while this work is a sub-set of that, focusing on how to model the knowledge needed. Let us move on to the solution suggestion – the models to model embedded configuration. That is the content of the next chapter.

# *Chapter 5*

## THE MODELS

This thesis suggests a method to help designers of mechatronic systems to deal with complexity of system setup where extreme postponement has been used. By extreme postponement we mean that no product variance is given in production. A black box is manufactured, and it is first in installation that a variance is selected. In this chapter, we describe in detail the three model types included in the method. It is necessary to remember that the evolution of the method is driven by a specific problem that is then generalized. At the outset, some issues were identified as compulsory in this method:

- The models to solve the specific problem of embedded configuration are a system comprising computer-controlled subsystems configured together to form a whole.
- It is necessary to be able to visualize a lot of information and still be able to make sense of things.
- Completely consistent semiotics are needed so programming can be done automatically. This has two consequences:
    - A complete model is also a completed programming.
    - Vice versa also holds, an existing program can be exported to a model.
- The same model is accessible to laymen, readable by CEOs, and detailed enough for professional modellers.

This may be an ambiguous start, but the elements needed for achieving this are out there. We should not forget the end reason for such a method: to simplify the work and life of the user. User here can be both the end-user and middle-users like maintenance personnel. The latter have suffered the consequences of complex multipurpose artefacts where *more is better* has been dominant for too long.

> *We must get rid of the psychology of 'more is better' and adopt Mies van der Rohe's slogan of 'least is most'.*
> *(p.618) in (Simon 2002)*

It is our belief that most users would like to have more for less, meaning that they wish to retain adequate functionality with minimum input from their side. To aid designers achieve such designs, we suggest the method: *Knowledge engineering for embedded configuration* or *Kefec*. The main purpose is to reduce complexity for users but to retain the same (or at least adequate) functionality of the artefact. The method consists of three models: *System-Breakdown Model (SBM)*, *Encapsulation Model (EM)* and *Communication Model (CM)*.

Before explaining these three models in detail, let us look at some concepts that are necessary to master for enhanced understanding of the models.

## 5.1 THE CONCEPTS

The method relies on four main concepts to implement its purpose. The concepts are *decision variables*, *phases, abstraction levels* and *relations*. These concepts aid in defining and breaking down the interaction process between the user and the artefacts with the aim of reducing complexity. The starting point is the user and the user's perception of the artefact. Where are the boundaries between him/her and the artefact? What crosses this boundary and why? Therefore, we want to define *decision variables* clearly, for the purpose of reducing them. Can we decompose the problem into more manageable parts so each becomes easier to solve? We introduce *phases* to deal with problem complexity, as we identify different environment complexity in different phases. How do artefacts relate to their functionality and purpose? Can we describe these with a predefined vocabulary, and even more, the relations between the levels? To deal with this, we introduce three *abstraction levels* that have to be defined for a more complete description of the artefact; and then elements in each level have to be related to each other, hence *relations*.

It is our hope that these four concepts can lay the foundation for tackling these questions and hopefully even provide some answers. Let us look at each concept, one at the time and see why they are here.

### 5.1.1 DECISION VARIBLES

A decision variable can be described as every "issue" or parameter for which a human has to decide the value. They can be trivial, like selecting a colour, or fundamental, like choosing where to live. Both are variables that require input from us humans. A graphic explanation is to use a system view to identify decision variables or DVs. Let us assume that a system has elements and relations between elements as shown in Figure 98.

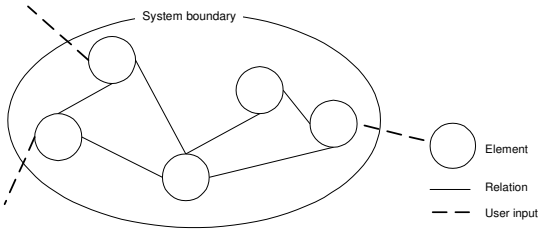



**Figure 98 - Decision variable transcends system boundary**

What happens within the system boundary is irrelevant to the user of the system. The inputs for which the user has to make decisions are the ones that are interesting. As we defined earlier in chapter 2.3 (page 20), all variables that break the system boundary (Figure 98) are called DVs (see Definition I: Decision Variables, page 21). The system boundaries can be for both subsystems and the overall system. To sharpen the focus, let us define three kinds of decision variables: *decision variable* (or *DV*), *tentative DV* and *internal DV*. The first, *DV,* breaks the overall system boundary and always requires input from the outside user, i.e. not only does it require input from outside the encapsulation model (EM) but also outside the complete system. The second, *tentative DV*, is a deeper layer of DV that only becomes "activated" if one chooses to "break" one (or more) of the defined relations. The third and last is the *internal DV*. This DV

requires input from outside the EM but can receive information from within the system. This is illustrated well by the example that follows.

To better understand DV let us look at a ventilation unit in a car. To control the ventilation, we know that there are several "parameters" we need to adjust to get what we need. To be precise, there are five parameters:

1. Blend of hot and cold air
2. Air-cooling on (AC)
3. Direction of the air blend
4. Only recycle air from within the car
5. Speed of motor that drives the air stream

We all recognize the interface, as shown in Figure 99.

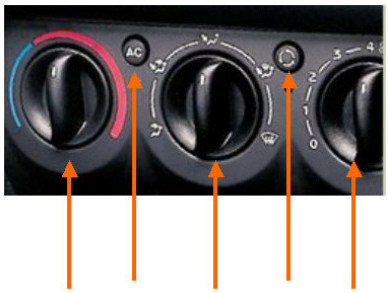



**Figure 99 – Interface for car cabin ventilation – five DVs**

We also know that these five DVs are not sufficient to achieve what we actually want; to get the result we wish for, we need to be constantly adjusting the values of these five DVs. These DVs are directly connected to the artefact construction and not the final customer need or application. In modern cars, manual ventilation control has been replaced with Climate Control as seen in Figure 100.

**Figure 100 - Climate control - One DV**

It is worth noting that suddenly there is only one DV: the temperature inside the cabin in Celsius. This DV is much closer connected to the actual customer need, as it relates to the fact that humans like the space they are in to have a constant temperature and that the comfort zone is between 18 and 26 degrees Celsius. This is a drastic reduction of DVs, from five to one. How can this be achieved? The simple answer could be:

*The closer one gets to actual customer needs, the fewer DVs are required.*

Right now, this is of course a hypothesis, but there are indications that it holds and that it is actually one of the main drives in suggesting the Kefec method. So our working assumption is that raising the abstraction level of DVs from artefact structure to customer need leads to a reduction in DVs. This is shown later on in more detail. Returning to the example, the transition from manual control to climate control is not a trivial one. It requires:

- Incorporation of knowledge

- Added artefacts to the system – here, most likely some sensors (outside, inside, and engine temperature).
- Encoded application knowledge, probably developed by observing usage for some time under different conditions and linking this to the available measurements, such as values from sensors. In other words, monitoring behaviour patterns and storing control plans to match.
- Keeping old DVs but adding new ones that are closer to the actual customer need, the application. This suggests that the method has to support DV layering.

This also shows that DVs can be of quite different types, i.e. speed of motor or temperature (in Celsius). Later, we use this same example to make a simple encapsulation model and show how to highlight DVs. An important aspect of this example is to point out the underlying rationale for making multi-abstract level models with rigid relations between elements. The whole premise of this thesis is to reduce the number of DVs! All aspects presented here aim at hugely reducing DVs.

### 5.1.2 ABSTRACTION LEVELS

Achieving simplicity is not simple. To be able to make more sensible DVs, it is necessary to incorporate different abstractions into the model. This is similar to the *data*, *information* and *knowledge* dilemma. In construction of artificial things, here named artefacts, these three levels become *artefacts*, *functions* and *applications*. The choice of these three names merits an explanation. Abstraction levels are not new; they have been suggested by many to describe things. In this thesis, the notion stems from domain theory (Andreasen 1980;Andreasen 1991), and it is later used in PVM (Hvam 1999;Hvam, Mortensen, & Riis 2007). The names of the levels have changed throughout time, and here we also choose new names in order to clarify and to remove some ambiguity. The lowest level is called *artefact*, in contrast to structure, product structure, component or parts. This is done first to free the word *structure* for use when talking about structuring, as in organizing or formalizing, and secondly to highlight the notion of whole instead of part or component. We also want each level to have single-word names for aesthetic reasons. The middle level, called *functions,* is just that; it has also been called engineering view. We allow functions to be described in different degrees of detail, so it can be said that the notion of *effects* (Hubka & Eder 1987) is included in this level. The top level is *application,* which links the artefact through its functions to the environment and hence the customer requirements. We wanted to use open, single-word names and found customer need, requirements and services not accurate enough or even misguiding. Application also hints at the software aspect of this work. The actual names as such are not important; what is important is that the reader understands what they stand for. Let us now look at the three levels in more detail, and see what they are about.

***Application*** is the highest abstraction in the model and has two main elements or pieces:

- Requirements, the wished state or performance, customer need
- Goals, the actual state or performance, system driven

We consider it necessary to distinguish between requirements and goals. The former is what the customer needs, while the latter is what the system can achieve. In an ideal

world, requirements and goals would be the same but this is hardly ever the case. It is therefore crucial to allow for discrepancies between the two.

***Function*** is the heart of the model: a functional description of the artefact in respect to the designer's intentions and based on the observable (and non-observable) behaviour of the artefact. It has two main groupings of elements:

- Intentional functions that contribute to goals, (may be call effect)
- Functions (technical, internal) that set hardware

But is it always possible to make this distinction? Probably not, but a good aid for identifying the two kinds of functions are Hubka's effects (see page 52) and Kitamura's functional concepts (page 78).

For a clear structure at the functional level, it is suggested to separate the intentional function from the basic ones. Intentional functions have internal relations to basic ones, while basic ones relate down to artefacts. Intentional function should probably not relate directly to artefacts. On the other hand, intentional function can be accessed from the "outside", from other artefacts, to be used in the overall system.

Several authors have described the process of moving from functions to the actual artefact. They all have in common that these are intermediate steps, and that the final description is in functions and the actual design. Two of the concepts that fit this are Tjalve's "principal solutions" (Tjalve 1979) and Hubka's organs (Hubka & Eder 1987). Knowledge of these methods undoubtedly helps in connecting the two levels, but it is by no means necessary. The reason for this is that both methods aim at new designs, while the primary focus here is logic restructuring with a known artefact structure.

***Artefact*** is the actual physical construction and is the only *real*, observable thing. This is the only level that can be affected directly by the designer. The other two are inferred from the artefact. We assume that artefacts are most often constructed of modules that are then made of parts. It is possible to "jump-over" the module level. The artefact is constructed in a hierarchical manner. It is also possible to add a level above the single artefact, a system level that is a kind of group of artefacts so a whole system can be handled in a single encapsulation model. This is described later in the thesis with the Stacked EMs.

This triple layer approach is to capture the purpose of artefacts; we could say that it is collecting information that was lost after the design was done. So the designer has to explicitly "document" his rationale from requirement, functions to the actual construction of the artefact. The customer's need and view of the artefact has to be mapped through functions to the actual construction. The decision variables mentioned earlier can be connected to each of the three levels.

All this is not new, as seen in the theory chapter. Thinking about artefacts at different levels of abstraction is done many places. What is added in this thesis is that these levels need to have rigid mappings, and their internal relations also have to be completely put in place. This leads us to the core of this thesis, relations!

### 5.1.3 RELATIONS

Relating things is something we all do all the time. This is also how our wonderful brain works; we *string us along* relations to search our brain when we think about all kinds of things. In that sense, it is rather strange that more effort has not been directed toward researching relations and their generality. Relations are also the core issue to be solved in making this modelling framework work. Introducing many abstractions without rigorously relating the levels in a complete way would only result in a feeling of bureaucracy and would probably be considered double (or triple) work.

Here, we introduce three groups of relations: *decomposition* of tree, *relations* that are internal in levels along with mapping between levels, and *system-artefact* relationships, which deal with communication between artefacts and hence between EMs. The latter two are both relations – what differentiates them is just whether they break sub-system boundaries or not. When an EM is made for each sub-system, the two types exist. If an EM is made for the whole system, only the former is present. In the terminology used here, the former is EM, and the latter is Stacked EM. For the sake of clarity, we keep them separate.

Here, we suggest some groups of relations. These are the overall groups possible in constructing relations. The list is rooted in the case observed, the relation used in the case, and is constructed from a variety of literature mentioned in the theory chapter 4.7.2 (page 86). Some relations are made with several instances of the same type of relations, since some relations are particularly interesting with regard to the problem at hand, and the researcher feels it necessary to differentiate them. The relation types are presented in Table 27.

**Table 27 - Relations groups and their relation types**

| **Relationship group:** | **Level / Place** | **Relations type** |
|---|---|---|
| *Decomposition* | All | Meronymic (part-whole) |
| | All | Classification |
| | Application | Requirement (intentional) |
| | Application | Goal (intentional) |
| | Function | The four behaviour types |
| | Function | Intentional function |
| | Function | Basic function (technical) |
| | Artefact | Module (spatial) |
| *Matrix Relations* | All | Causal |
| | All | Intentional |
| | All | Temporal |
| | Artefact int | Spatial |
| *System artefact* relations | All | Case-role |
| | All | Causal |
| | All | Intentional |
| | All | Hardware (intentional) |
| | All | Application (intentional) |
| | All | Temporal |
| | Artefact | Spatial |

An observant reader might ask: What about the practical types introduced by (Li et al. 2007)(see page 91)? Most of them are actually represented, for example, by just having the matrix F-S one has given room to *has_function* relation. Another can be represented by entries in the PVM; e.g. *has_material* is just an attribute of an entry. Only relation types *has_process* and *use_material* are problematic, as they are not properties of the artefact but of the external system, outside our scope. Back to our suggestion – we deal with each group and its types in detail where the relations are used; decomposition and matrix relations in the encapsulation model; and the system artefact relations in the communication model sections.

One of the most important aspects of the relations is their visualization. The three groups are visualized in three ways. The first, decomposition, is shown as a tree. The second, internal and mapping relations, is shown as matrixes where each relation type (and subtype) has a certain symbol (or colour). The last group, system artefact relation, is the hardest to visualize. This is largely because an unknown number of artefacts in the system can lead to a different system picture each time. Artefacts have functional roles but very often a redundancy exists in the system. These variances are not without limits, however; this happens within a specific domain for which it is possible to make "guidelines". We use a multi-graph, a modified parametric diagram, a tree list and ontology to visualize system artefact relations.

### 5.1.4 PHASES AND INDUCED CONFIGURATION

When observing how multipurpose artefacts, which rely on internal software to select variants, are combined to form a system, a clear distinction can be seen in the activities that are performed. These can be grouped into four phases as shown in Figure 101.

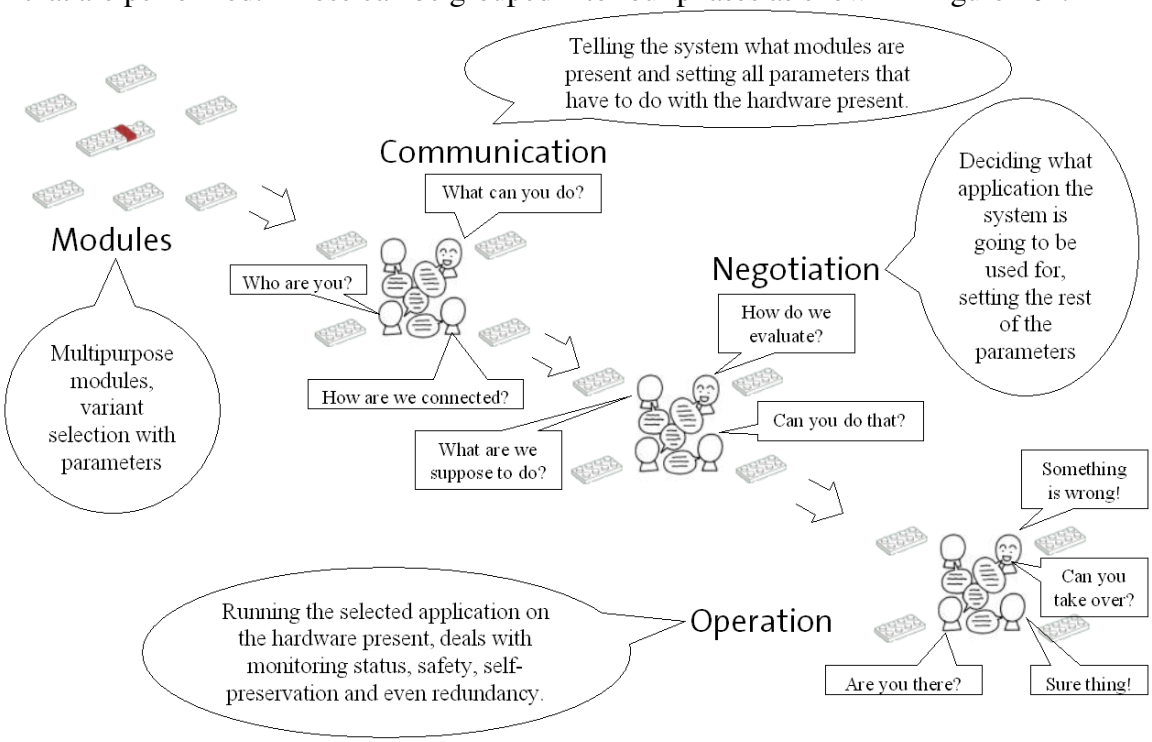



**Figure 101 - Phases in multipurpose modular system setup**

Let us look at these four suggested phases and see what happens in each of them.
Phase 1 - "Modules"

Modules need models (knowledge / information) on its artefact structure, functions, how these tie together and how these can be offered to others as services (feature, application). This leads to a multi-view product model and method for structuring data for encapsulation. Included in the model should be all the parameters needed to set each module and what parameters are to be accessed from other modules (encapsulation).

Phase 2 - "Communication" or *Setting the solution space*
Once modules are physically connected, they need to eliminate "illegal" solutions. This phase is about what can the system do, who does what, is there a redundancy in the system and if so, who has priority and so on. The whole purpose is to set / define a solution space from which one solution can later be selected. This is a precondition for setting an application, like constructing the rules needed to make a legal solution in the configuration. If knowledge and communication are structured to a satisfactory degree, this phase could be completely automated, as all necessary knowledge could be encapsulated into the modules. We like to call this "*Hardware-induced configuration*" or H-I-C.

Phase 3 – "Negotiation" or *Selecting a legal solution*
After the hardware has been coupled together, a specific solution can be selected to meet the required application. This is based on some input from other systems, humans, interpretation of customers' requirements and so on. But in essence, this is selection of one legal solution, probably out of thousands, tens of thousands or even millions of possibilities. The selection uses some kind of evaluation model to decide on best solution. The evaluation can be constrained by the input, e.g. customer wants constant pressure system. We like to call this phase "*Application-Induced Configuration*" or A-I-C.

Phase 4 – "Operation" as in operation of system
The selected solution is then put into operation. Along the way, some things can change, e.g. one module breaks down, detects it itself, and tries to move its functionality over to the other modules. Another very important aspect of this phase is its self-protecting functionality, i.e. each module tries to protect itself from being destroyed.

The reason for introducing these phases is because they exhibit different environmental properties. A fine way to classify environments is suggested in AI. It builds on looking at PEAS (Performance measure, Environment, Actuators, Sensors) for the system and determines five binary statements. The statements deal with observable, deterministic, episodic, static and discrete properties of the environment. In the case, it was observed that the phases showed some different properties. The result of evaluation of the task environment for a fresh water pump system is shown in Table 28.

**Table 28 - Task environment, as found in the case company**

| Task environment / Phases | | | | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|
| ***1*** | ***F***ully observable | vs. | ***P***artially observable | F | F | F | P |
| ***2*** | ***D***eterministic | vs. | ***S***tochastic (and strategic) | D | D | D | S |
| ***3*** | ***E***pisodic | vs. | Se***Q***uential | E | E | Q | Q |
| ***4*** | S***T***atic | vs. | D***y***namic | T | T | T | Y |
| ***5*** | ***D***iscrete | vs. | ***C***ontinuous | D | D | D | C |

Let us discover a single question to aid in deciding whether task environment is one or the other on the five binaries.

1. Can the agent (or agents) observe the complete state of the environment at all times? (yes => F)
2. Is the next state of the environment COMPLETELY determined by the current state and the actions executed by the agent(s)? (yes => D)
3. If an episode includes the agent to perceive and take one action, is the next episode independent of the actions taken in the previous one? (yes => E)
4. Can the environment change while the agent is deliberating his actions? (yes => Y)
5. Can the problem be viewed in states? Is time not continuous? Are there discrete sets of perceptions and actions? (all have to be yes => D)

Much more on how to classify environments is to be found in (Russell & Norvig 2003). The whole purpose of this is to determine level of complexity in the problem at hand, and sequentially how to solve it. Task environment can also be used to explain why we suggest splitting the installation process into four phases. Rationale behind this is based on the "score" from task environment shown in Table 28. Phases one and two are more controllable than phase three. Both one and two have a simple "score", and three is sequential (Q in question 3). A quick look at phase four in Table 28 shows that it has the highest score in complexity. So, the first three are quite different from the last phase. Or to rephrase this, we "push" the complexity away from phases one through three and into four to make it easier to solve the problems.

It is also recognized that the phases do not have the same properties in all problems. The working hypothesis is that only phase four will change, the other three will still show these properties (Table 28). This is also why phase four is excluded in this modelling framework.

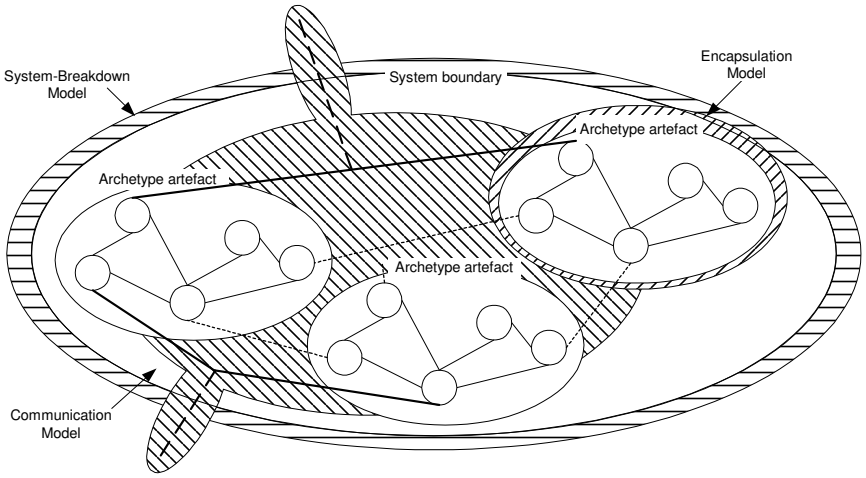



**Figure 102 - The three models of Kefec**

## 5.2 MODEL SUMMARY

The modelling framework suggested here consists of three models or groups of models. These models are interlocked and present different aspects of modelling systems that can support embedded configuration. The models are: *the System Breakdown Model*

(SBM), *Encapsulation Model* (EM) and *Communication Model* (CM). The SBM describes the overall system, its main artefacts and separation of the system and its environment. The EM describes a single artefact in detail in three abstraction levels, and finally the CM describes how elements in the EM are formalized and how artefacts in the SBM can exchange information (communicate) through the EM. The models and their overlap are shown in Figure 102.

The purpose of the three models is to give the modeller a tool to aid in the design and redesign of systems that rely or could rely on embedded configuration. Let us look at the three models in the order they are presented here, starting with the SBM.

## 5.3 THE SYSTEM-BREAKDOWN MODEL

The system-breakdown model or SBM (also called SBS, *system breakdown structure* in other methods) is the general domain-dependant description of the system in question. The purpose of the SBM is to show the boundary between the system and the environment and to pencil out the overall expected purpose.

### 5.3.1 ARCHETYPE MODEL

To describe the purpose, we like to introduce the concept of *archetype*. An archetype is an organ (Hubka & Eder 1996) or multi-organ where an artefact (or part thereof) is describe from its functional aspect. Archetype is a typical subsystem present in the system and its functionality should be described with functional concepts or intentional functions.

In the SBM, a description of the archetype subsystems within the system is made; an evaluation of the environment is carried out in the section, *Phases and Induced configuration,* on page 115. It is worth mentioning that the environment is not the same in all the phases. The outside of the system boundary has been pushed to phase four (plus a small part of phase three). So, the environment evaluation applies mainly to phase four, but the others have to be confirmed.

The system is to be described in the domain lingua of the problem at hand. It is preferable to use artefacts in the form of archetypes to describe the system. An example is shown in Figure 103.

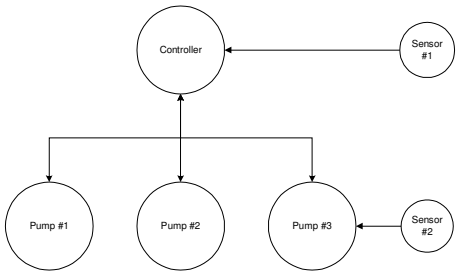



**Figure 103 – Archetype model for a fresh water supply system**

Do this graphically, with a simple non-standard method as in Figure 103 or use any modelling standard that is preferred within the company. Keep the archetype model on a rather high level and in a form that domain experts can understand.

To help find the archetypes, the PEAS method can be deployed. It helps to categorize the artefacts and hence identify the relevant archetypes. It is made in a table, like the one presented in Table 29.

**Table 29 – PEAS description of task environment**

| Agent type | **P**erformance measure | **E**nvironment | **A**ctuators | **S**ensors |
|---|---|---|---|---|
| Water supply system | Water pressure from tab | Pumps, pipes, controllers, sensors, valves | Motor speed | Flow, pressure |

The purpose of the archetypes is to generate an overview of the domain in which the systems are later constructed.

### 5.3.2 FUNCTIONAL STREAM MODEL

Functional stream model is a general description of the system in a functional language. The model should be based on the archetypes, and we should describe in words the purpose of the system, the widest combination of archetypes. From that, we can make single row flows of the purpose of system. Draw the streams individually, even if it means repeating items in several streams. The result is called: *Functional stream model*. The modelling language can be based on Functional Basis or MFM, but the items have to be described with functional concepts (high-level functions or effects). Note that FSM is drawn for the WHOLE system and not for each subsystem.

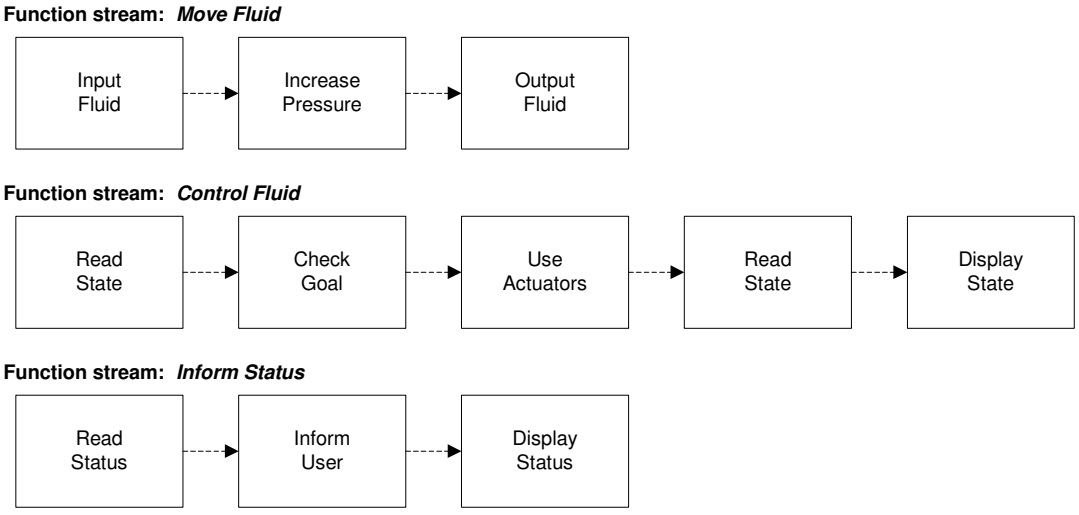



**Figure 104 - Functional Streams for case, made by researchers**

To clarify, let us describe and draw the functional streams for the case company product, a pump system. There are three main functional streams: *Move Fluid*, *Control Fluid* and *Inform Status*. In addition, there are some aid functions, but they are not relevant here. Within these three streams, there are several functional concepts. Figure 104 shows these procedures in a FSM. Even though these streams are drawn

individually, they can actually overlap and interlock. For example, *Inform Status* could be alarms, where *Control Fluid* is invoked to stop the system in case of emergency, and so on. Some of this overlap and interlocking mechanism can be drawn in the SBM later on. The purpose of the FSM is to highlight the main functionality of the system, to visualize the streams, and finally, to allow for later breakdown to map archetypes in the SBM.

At a workshop with the case company (presented later in test #4, page 149), a slightly different view was reached. This resulted in four functional streams where none was focused on actually moving liquid. This became a hidden action within "actuate". The resulting functional streams are shown in Figure 105. For clarity, let us use the workshop results for the construction of the system breakdown model.

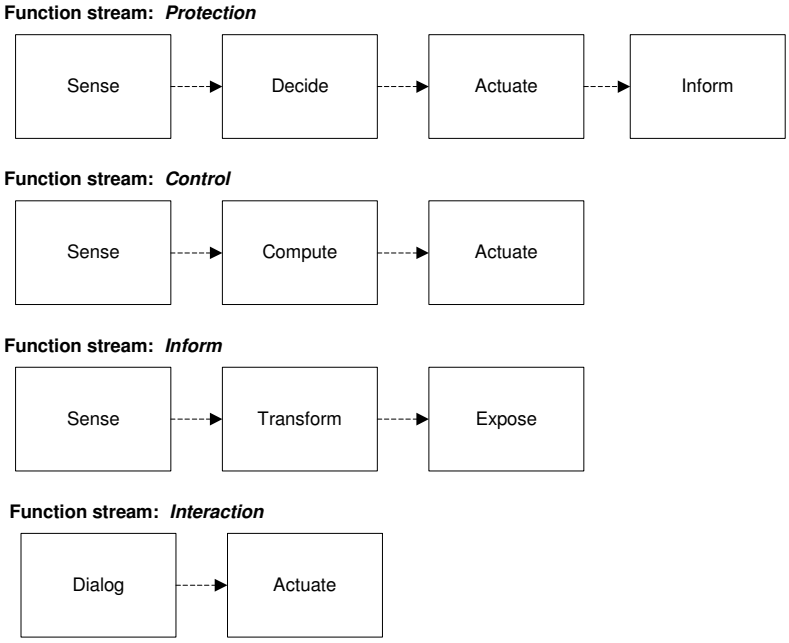



**Figure 105 - Functional streams for water supply system made by case company**

The two groups of functional streams are shown here on purpose. It is to highlight the fact that two groups of people can reach two different ways of describing the same system. With iterations and negotiation, it is probably possible to reach consensus but this requires some work. This might be the main drawback in using functional descriptions when describing artefacts. Having this in mind, the next step is to draw the system, its archetypes and its boundaries.

### 5.3.3 THE SBM

The first two models mentioned in this chapter are intermediary steps to guide the construction of the main model, the System-Breakdown Model or SBM. By overlaying both the archetype model and the functional stream model, we are able to generate the SBM model. In the description, it is necessary to identify the inputs and outputs in the form of archetypes and connect them to the archetype artefacts identified earlier. So, the SBM model is basically an archetype model with system boundaries and functional concepts as elements within each archetype. It should point out inputs and outputs through lines that break the system boundary. A SBM model is shown in Figure 106.

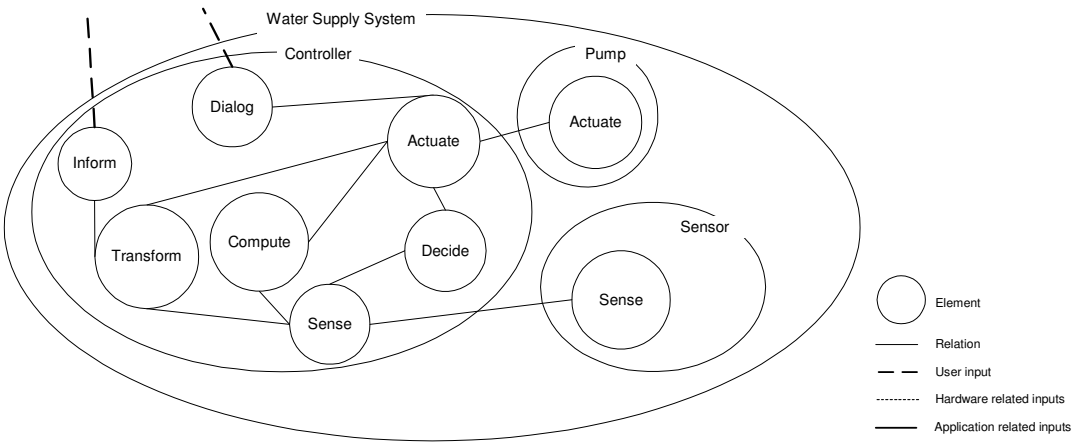



**Figure 106 - SBM model of a system**

With the SBM in place, we can start to evaluate the environment outside the system boundary. The aim here is to estimate how complex the problem is. The system in question could be a part of an even larger system, or it could be a stand-alone. To evaluate this, we look at a task environment evaluation for the operation phase, as that is the phase not shielded from the outside. We suggest using the five binaries in Table 30 to categorize the outside environment.

**Table 30 - Task environment evaluation**

| | Task environment / Phases | | | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|
| ***1*** | ***F***ully observable | vs. | ***P***artially observable | F | F | F | |
| ***2*** | ***D***eterministic | vs. | ***S***tochastic (and strategic) | D | D | D | |
| ***3*** | ***E***pisodic | vs. | Se***Q***uential | E | E | Q | |
| ***4*** | S***T***atic | vs. | D***y***namic | T | T | T | |
| ***5*** | ***D***iscrete | vs. | ***C***ontinuous | D | D | D | |

Others have given measures on complexity and environment, for example the summary in (Jensen 1999), page 59, where the author draws from Ackoff, Blanchard & Farbrycky, Hubka & Eder and Klir & Valach. Most of the variables in Table 30 are presented there, only not is such a complete manner.

The task environment evaluation is taken directly from AI work (Russell & Norvig 2003) and is good because it helps us to determine if our system is possible for Kefec treatment. If binaries in lines 1,2,4 and 5 in Table 30 are not the former letter (F,D,T,D), it is advisable to be VERY aware of whether the system can be made "smarter". The value in 3 can be both E and Q without causing problems. These five binaries describe the environment sufficiently to aid in evaluation of the system environment. To "prove" this, we point to the AI work of (Russell & Norvig 2003), chapter 2.3, page 38, since the discussion is lengthy. The last part is to deal with the limits and constraints of the system. It is advisable to identify and state what archetypes have to be present, what is optional, and other information needed to better understand the system.

A lot of the work needed here is actually requirement formulation. We have no intention of inventing a new method for such work and point the user towards using some of the excellent standards already in place for doing this. Methodologies like Systems Engineering (SE or ISO 15288) or Requirements Engineering (RE) are perfect

candidates, and are directly applicable to the task of formulating the SBM model and its boundaries. The archetypes presented in the SBM model point out how many encapsulation models are needed, since one EM is to be made per archetype. Let us look at the EM and its definitions.

## 5.4 THE ENCAPSULATION MODEL

For each artefact in the SBM model, we have to model in detail what happens with it. The encapsulation model could be called Product-Breakdown-Structure or PBS as that is its function. The name *Encapsulation Model* (*EM*) is chosen to highlight its greatest goal, to encapsulate as much information as possible. The model is made of a centrepiece with three groups and six matrixes to connect the three groups. The matrixes on the left are internal relations or are also called intra-domain matrixes (Malmqvist 2002), where both rows and columns have the same elements. The matrixes on the right are mapping matrixes or inter-domain matrixes. In this case, it can be said that the domain is an abstraction level. The centrepiece is a tree describing the artefact on three different abstraction levels: application, function, and the artefact itself. The encapsulation model can be seen in Figure 107.

The point in this model is to generate a multi-view model, seen from afar. The details blur out and only patterns are seen. Looking more closely, the details appear and individual elements and relations can be traced. The rationale behind this involves the magical number seven (Miller 1994), the fact that the human mind is capable of handling seven plus-minus two *chunks* at a time. From afar, a single matrix becomes a chunk, a pattern, which the mind is very good at dealing with. A whole model with really many items in the tree and up to several thousands relations can be reviewed at a glance. We illustrate this attribute of the model in much more detail later (see page 181 and onwards).

Let us now look at each chunk of the model: the centrepiece, the internal matrixes and the mapping matrixes, in that order.

### 5.4.1 THE CENTERPIECE

In the middle of the encapsulation model is the centrepiece, a tree structure that lists three abstraction views of the artefact. This tree is based on the PVM (Hvam, Mortensen, & Riis 2007). What has been added here is a stringent notation for the symbols used, what they mean and how to go about making the decomposition.

**Figure 107 - The encapsulation model**

The centrepiece is shown in Figure 108.

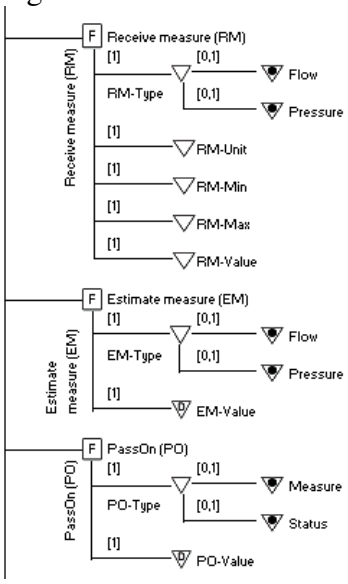



**Figure 108 - The PVM in the centre of the encapsulation model**

The centrepiece has a lot of explicit and implicit information coded into it. A walk through is in order, so let us start by looking at the tree itself, then the symbols used, the relations that are shown in the tree, and finally the additional information in the model. The tree is to be made formal in a special grid. It has levels and branches as shown in Figure 109.

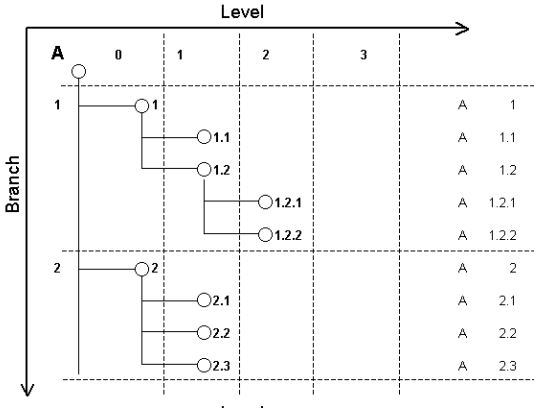



**Figure 109 – Levels and branches in the PVM tree**

The branches are used for groupings, like a module, and they move from top down in vertical manner. They are still meronymic relations and are the first decomposition of the abstraction level. Each branch has a number, starting from the top. The levels are also meronymic relations in a more conventional part-whole way. They show the parts of the module in a tree-like level structure. The levels are also numbered, starting from left to right. The top level is level zero and coincides with branch number. Levels are used to determine the details in the tree model. Assuming that we want to model everything, we can decide what to show by going down to level 1 or 2, etc. Each node in the tree gets a unique name in this manner, where the first number is the branch and each sequential number is a sub-branch number within this level, separated with a dot (see Figure 109). Classification is shown with a hyphen. A complete identifier is shown in Figure 110. To separate the abstraction levels, three letters have been chosen, *A* for application, *F* for function and *S* for artefact.

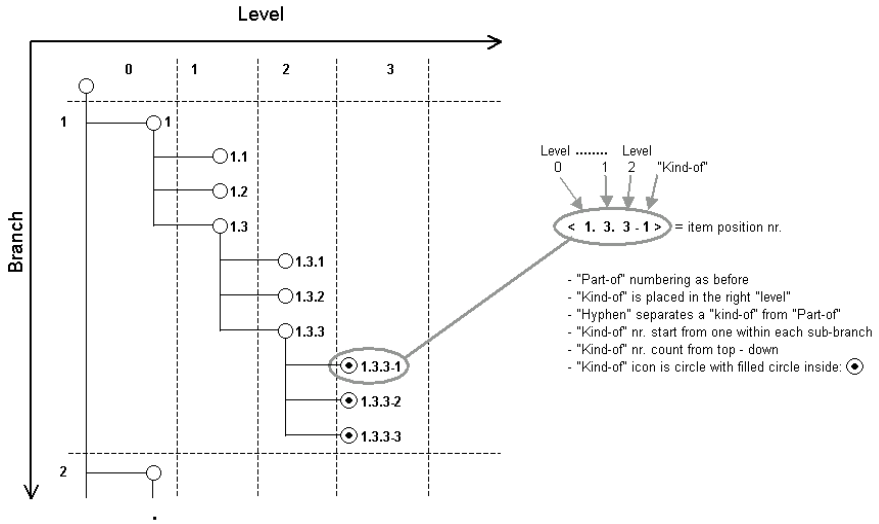



**Figure 110 - Identifying nodes**

This might seem a little trivial, but there are two reasons for this: first, to allow for control of the details shown, some sort of presentation rules have to be laid down; and second, a unique reference to each node is needed to establish a programmable model.

A complete syntax for programming the relations should be suggested. Having this in mind, we decompose the artefact is such a way that the same level holds roughly the same information. This allows for relevant hiding and showing of details. Another thing to remember is that relations probably influence how the decomposition is done. But there is more on how to actually construct the model in chapter 7.2 and onwards.

The model is quite focused on dealing with black boxes where variance is made with software. This means the symbols used in the tree are quite influenced by this. There are three main groups of symbols: a circle for parts, inverted triangle for software parameter, and square for higher abstractions, both function and application. A complete legend can be seen in Figure 111.

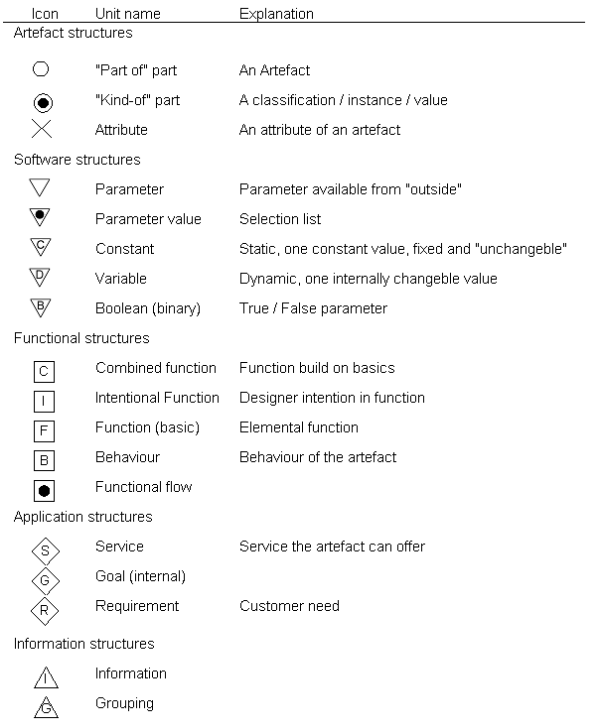



**Figure 111 - PVM legend**

When combining symbols to form a tree, relations are drawn. It is possible to put several different relationship types in place in the tree itself. These relation types are shown in Table 27, and combinatory symbols show them as seen in Table 31:

**Table 31 - Relations in tree structure**

| Symbol | to | Symbol | Relation type |
|---|---|---|---|
| ○ | – | ○ | Meronymic |
| ○ | – | ◉ | Classification |
| ▽ | – | ▼ | Classification (instance) |
| ▽ | – | ▽ | Grouping (Intentional) |
| F | – | F | Grouping (Intentional) |

To facilitate communication, which we introduce later, some additional information needs to be coded in the tree. Information is coded both to the left and right of the tree structure itself.

On the left hand side is information on decision variables and quality. An evaluation is placed in front of the tree on functions, stating the *quality* of their performance. An example is shown in Figure 108, where the function, *receives measure,* is rated 1, while *estimate measure* is 2. This is to be read thus: if the former is available, it is to be used, only to fall back on the latter, if the other becomes unavailable. This is an essential feature, as many black box artefacts have redundancy functions. The other information placed in front is the decision variable. The idea here is to show clearly what variables have to be communicated outside the EM, both to other EMs and outside the overall system. These are called *Internal DVs* and *DVs*. The decision variables can be layered into DV and *Tentative DV*, an explosion into details. $DV_T$ can be thought of as comprising items in the group of DV. $DV_T$ is usually controlled by DV, but access can be given if necessary (see example below in Figure 119). Table 32 shows the information coded on the left-hand side.

**Table 32 – Left-hand information panel**

| Information | Values | | Description |
|---|---|---|---|
| Decision Variable | DV | | Variable that needs input from OUTSIDE the system |
| Tentative DV | $DV_T$ | | A lower level DV, normally controlled by DV but access can be given directly (see Figure 99 and Figure 100) |
| Internal DV | $DV_I$ | | Variable that needs information from another EM |
| Quality | Q | | Numeric value for quality of function, 1 is best |

On the right-hand side, there are five information boxes.

**Table 33 - Right-hand information panel**

| Information | Values | | Description |
|---|---|---|---|
| Default | D | | Default value |
| | $D_L$ | | Last selected is the default value |
| Induced-Configuration | H | | Variable has to do with hardware setup (solution space) |
| | A | | Variable has to do with application (selection solution) |
| Information storage | I | | Information storage is internal |
| | E | | Information storage is external |
| Communication initiation | S | | Communication is initiated by the system |
| | E | | Communication is initiated externally |
| Function-group | 0 | | Change of an attribute value of an operand at the same location over time |
| | 1 | | Change of an attribute value of an operand from the input port to the output of the device. |
| | 2 | | Change of something inside of a device rather than input/output ports. The 'something' could be motion of a part of the device or inner state of the device |
| | 3 | | Any behaviour to another device. |

These are constructed to aid in the process of finding DV and to map the communications needed between sub-systems. It is probable that Table 33 has a lot of redundant information. Most likely, four of the five information boxes (all except *Default*) are only work-in-progress to construct the three DVs. These four are shown here for the sake of clarity and to test the relation between them and the DVs.

A pump is seen in Figure 112 where different types of information are coded to different functions.

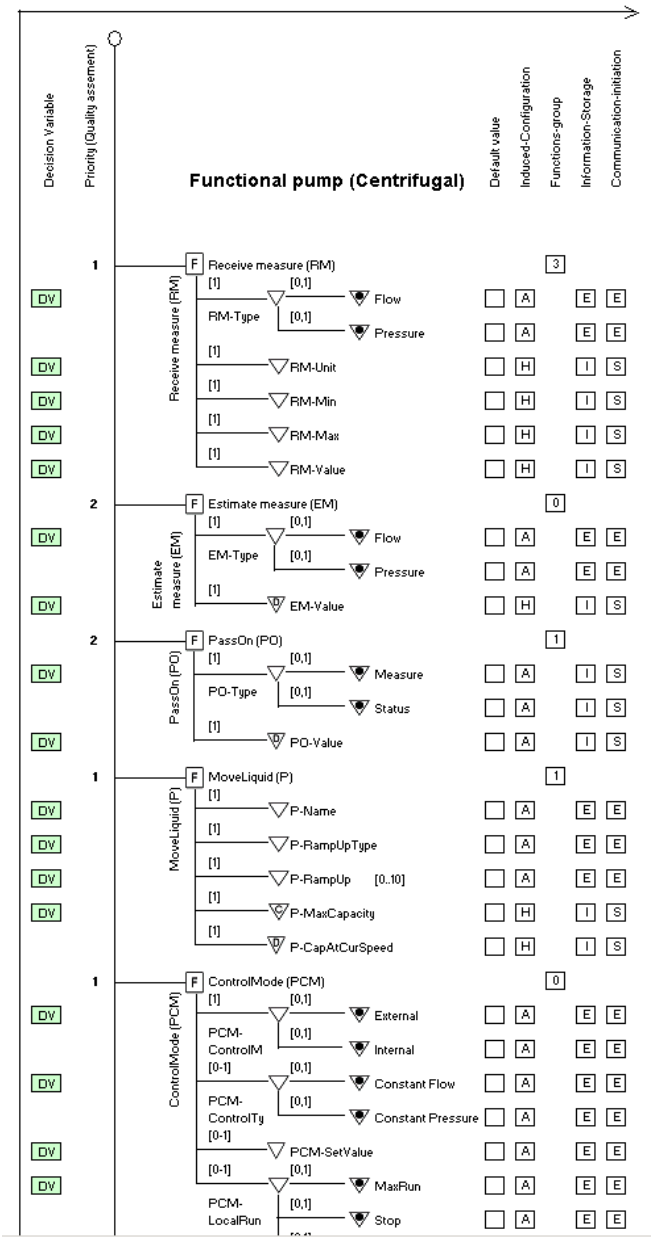



**Figure 112 - Additional information in tree**

An important issue here is the induce-configuration. We saw that a lot of parameters have to do with what hardware is present, or with setting the solution space. We call this *Hardware-induced-Configuration.* When a solution is selected a lot of parameters have to be set to make it work. We call this *Application-induced-Configuration*. The reason for placing this information here becomes clear when we discuss the decision variables and the communication model.

### 5.4.2 THE INTERNAL RELATION MATRIXES

Now that we have looked at the centrepiece, it is time for the main focus here, the visualization of relations. There are two kinds of matrixes: internal, where the same elements are in rows and columns; and the mapping, where different elements are in rows and columns. This corresponds to intra- and inter-domain matrixes (see Figure 91, page 104).

An element in an abstraction level can have relations to another element in the same abstraction level. These relations are shown in the internal relational matrix to the left of the centrepiece. Internal matrixes thus have the same elements on the vertical and horizontal axis. The elements run from the top down in the vertical and from right to left in the horizontal. This makes a *no-relation-possible* diagonal running from top-right to bottom-left as shown in Figure 113.

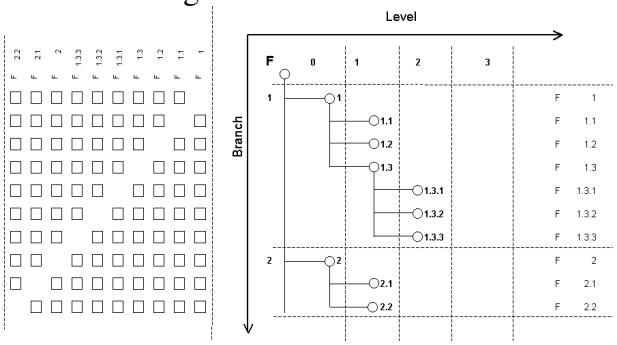



**Figure 113 - Notation of relational matrixes**

The node number can be used on both axes, allowing for programming of the relations. A box in the matrix is a possible relation between the two elements. These boxes can be populated with several symbols, depending on what kind of relation exists.

Possible relations are presented in Table 34. These are constructed from analysing the product data handed over from the case company and not by using the literature. The list here was enough to put all relations in the product in place in the model.

**Table 34 - Relations legend**

| Name | Symbol | Colour | Description |
|---|---|---|---|
| True | T | | The proposition is true |
| False | F | | The proposition is false |
| Not needed | $F_N$ | | The variable is not needed or is set internally (dynamically) |
| Exist | $\exists$ | | One or many variables exist => Exist at least once |
| Exist | $\exists!$ | | The variable/value exists => Exist only once |
| Optional | $O_i$ | | One variable must be chosen from group i = Exist one (in group) |
| Needed | N | | Has to be there |
| Calculation | $C_i$ | | Calculation on the variable in the group i |
| Bigger than | > | | The variable value must be bigger than the value of the testing variable. |
| Bigger than | >## | | The variable must have a value bigger than written #. |
| Smaller than | < | | The variable must have a value smaller than written. If nothing is written, then the variable must be smaller than the value of the testing variable. |
| Equals | "=" | | The variables are equal to each other |
| Value | V | | The variable has a specific value |
| Lowest | $L_i$ | | Lowest value in group i |
| Highest | $H_i$ | | Highest value in group I |
| Present | P | | A relation is present on some member of group |

Comparing these to the relations suggested in section 5.1.3 Relations on page 114 shows that these are all of the same type: intentional. For the purpose of analysing the product, we introduce colour coding to facilitate quick visualization of the cascading effect. The colour coding is as presented in Table 35.

**Table 35 - Colour coding of relations**

| | | |
|---|---|---|
| | Green | Proposition is true, this value selected automatically |
| | Red | Proposition is false, this value selected automatically |
| | Blue | One or many variable have to be selected manually |
| | Pink | Artefact (or part) has to be there or variable selected |
| | Orange | Calculation across variables |
| | Yellow | Logical propositions |

The thought here is to be able to very quickly find loose ends. These are variables that do not connect to an overall goal or application and have to be set manually. Blue in the matrixes is "loose ends" and can be present both in internal and mapping matrixes. The colours serve to make the *palette*, to be presented in a later chapter.

The relations can be combined in a logical way, e.g. by using propositional logic. It is the negation, conjunction and disjunction that are of interest. The first-order logic is used directly as relations (see above).

| | | | Symbol |
|---|---|---|---|
| Propositional Logic | Negation | Not | $\neg$ |
| | Conjunction | And | $\wedge$ |
| | Disjunction | Or | $\vee$ |
| | Implication | Implies | $\Rightarrow$ |
| | Biconditional | if and only if | $\Leftrightarrow$ |
| First-Order Logic | For all | | $\forall$ |
| | Exits | Exist at least once | $\exists$ |
| | Exits | Exist only once | $\exists!$ |

When a relation has been identified, it can be placed in the matrix. A typical IF-THEN statement that is very common in relating parameters is put in the matrix with IF in the vertical and THEN as the horizontal as shown in Figure 114.

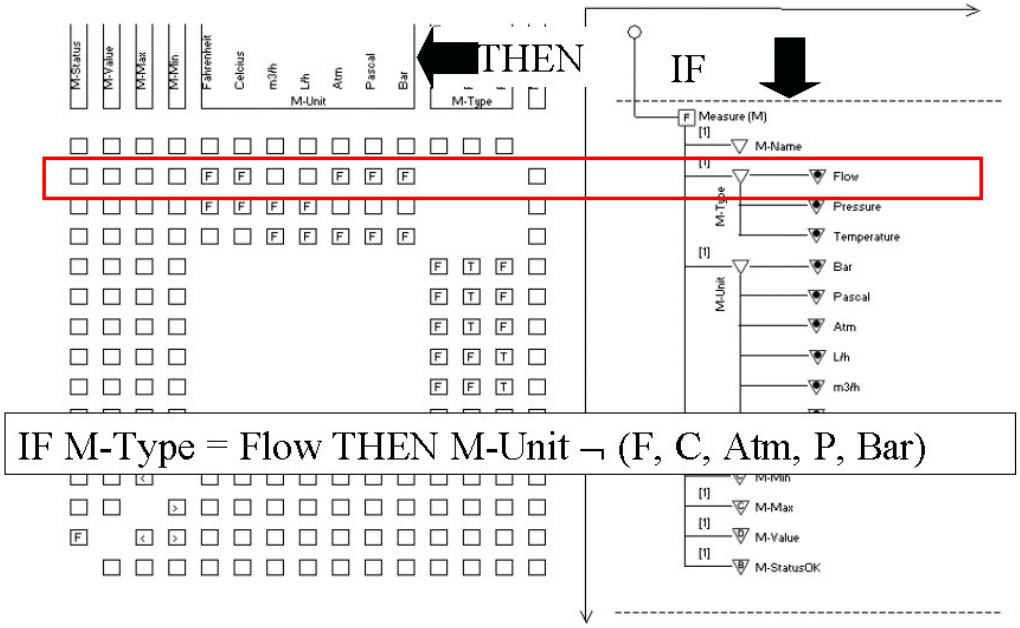



**Figure 114 - Internal relation matrix**

Blanks in the matrix mean that there is no relation possible. This happens when classifications are shown in the tree, like parameter values. When the internal matrix is completed, it should show how related a single abstraction level is to its own workings.

There is much more on the evaluation of the matrixes in chapter 7.6. The other matrix set in the encapsulation model is the mapping matrix

### 5.4.3 THE MAPPING MATRIXES

The mapping matrix shows relations between abstraction levels. It is placed on the right-hand side of the centrepiece. The main objective of mapping is to introduce a formal way of linking higher abstractions to a concrete artefact, module or part through functions. The final purpose is to allow for identification of DVs on the application level that will cause a cascading effect through the model and allow a single DV to set a lot of other parameters.

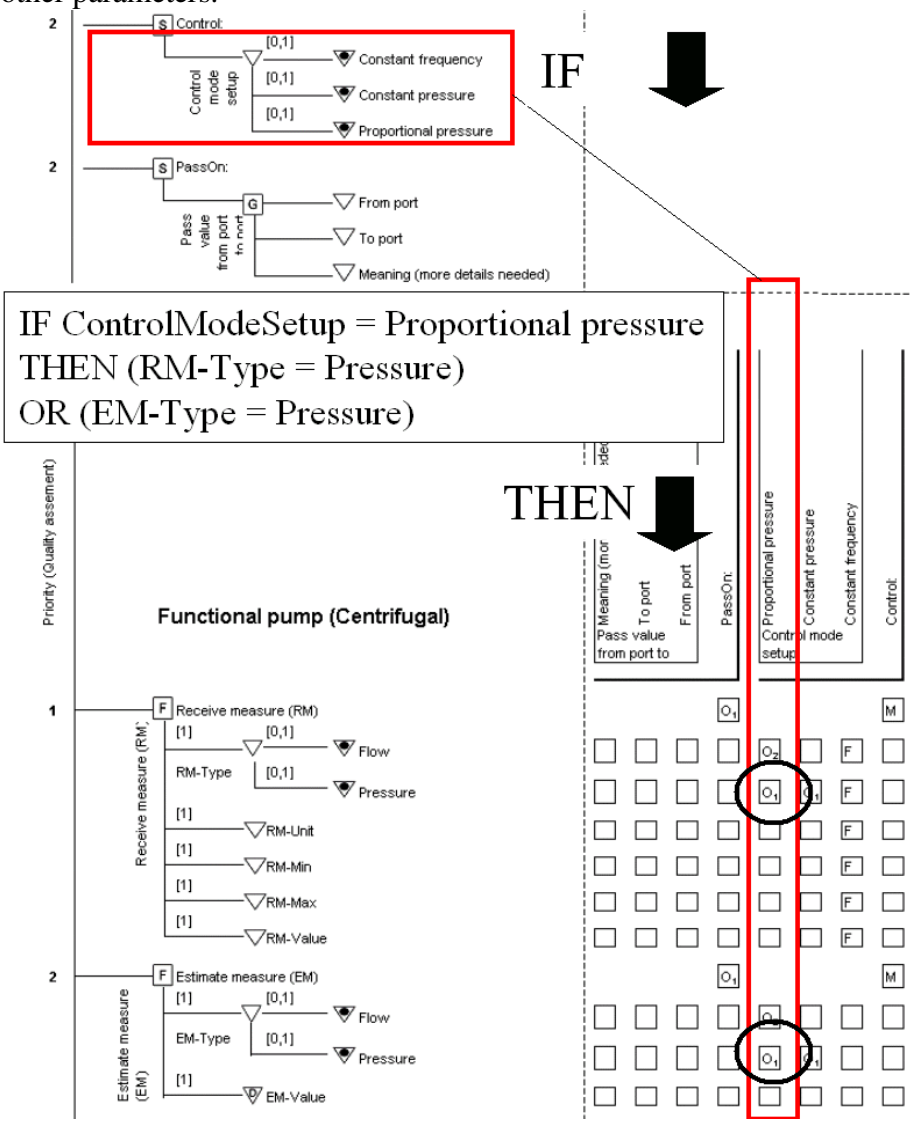



**Figure 115 – Inter-level mapping matrix**

Reading the mapping matrixes is the reverse of the internal matrixes. As the mapping is to connect top-down, the matrix is read with IF on the horizontal to the THEN on the vertical, as shown in Figure 115.

The notation used in the mapping matrixes is in the inter-domain style, where the rows and columns have different elements. The top abstraction is “dropped-down” to the columns, keeping the sequence through the drop so that the top element in the tree

becomes the last column and vice versa. The element numbers get the abstraction letter in front in order to differentiate the elements, as shown in Figure 116.

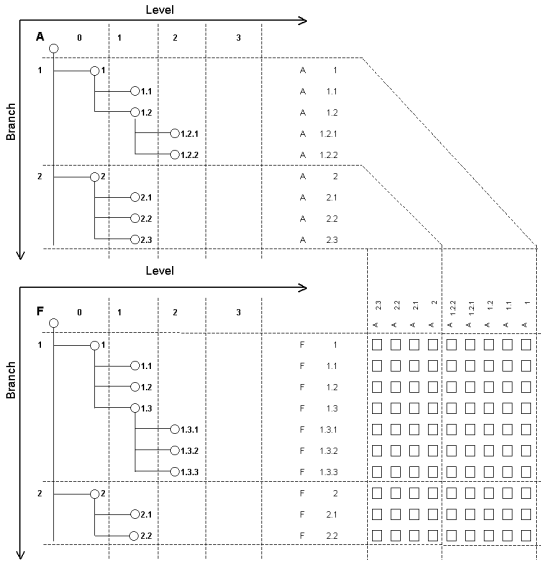



**Figure 116 - Notation in mapping matrixes**

Returning to the relations, all of the relations that can be used in the internal matrix also apply here. In addition, there are several other relations that are only to be used in the mapping. These are as follows:

- *States*: It is possible to introduce an application on the top level that describes a state. It can then set a group of variables through the mapping matrixes.
- *Case-role*: Similar to state, introduces a higher abstraction, which then cascades through the ranks.
- *Intentions*: Intentional relations are the most used here, but intentions can also be introduced in the form of entry on the application level.

All these have in common that they introduce an element in application that then allows relations to be coupled to it throughout the matrixes. Once all the mappings are in place, it is possible to evaluate how connected the abstraction levels are. But in order to do so properly, all the variables that have to be set have to be highlighted. These are the decision variables or DVs. Let us look at how to visualize them.

### 5.4.4 DECISION VARIBLES IN THE MODEL

The whole purpose of the encapsulation model is to "hide" information from the "outside", both from other parts of the system and the whole system. To do so, the idea is to describe the artefact in three abstraction levels and then relate elements in all levels together. The decision variables are then defined by analysing what variables need inputs from outside the model. It is therefore very important to highlight DVs in the model. As discussed earlier, DVs are probably layered. On top, we have the final DVs that are possibly constructed from a set of tentative DVs. Such structure allows *diving* into a different DV layer; remember the example earlier with climate control in a car, one manual and the other automatic, but still with all the manual parameters accessible.

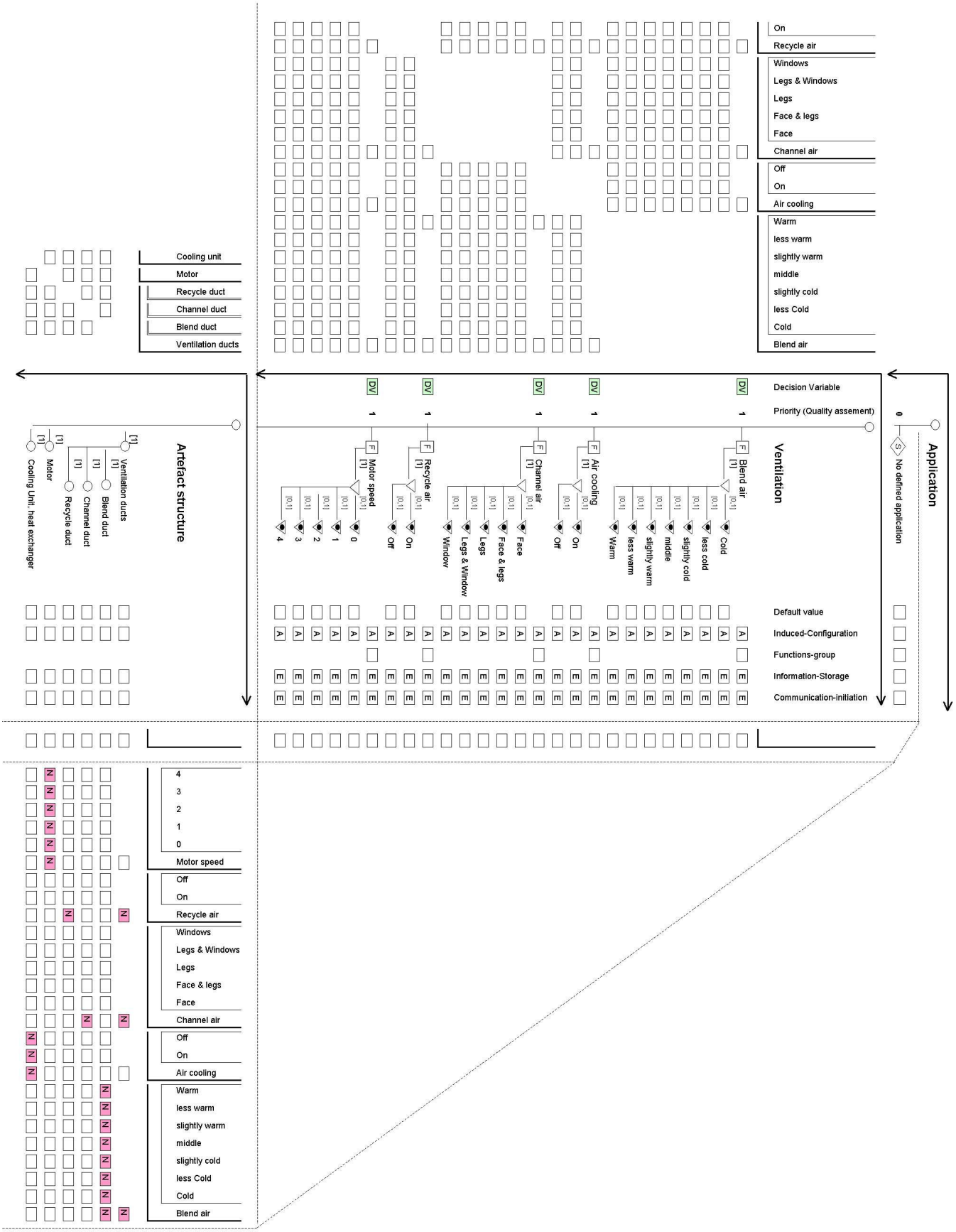



**Figure 117 - Manual ventilation in an automobile (HVAC)**

To show how decision variables are used in a model, let us look at exactly that example drawn from the real world, an air-conditioning unit for a car, often called *Heating, Ventilating and Air-Conditioning* unit or HVAC. An example of such units is shown in Figure 99 and Figure 100. Let us model these two units and see how the decision variables fall into place. A complete model of a manual HVAC, as shown in Figure 99, could look like the model presented in Figure 117.

Because of lack of relations, both internal and mapping between values, it is not necessary to show a fully exploded model view, and the collapsed version shown in Figure 118 will suffice.

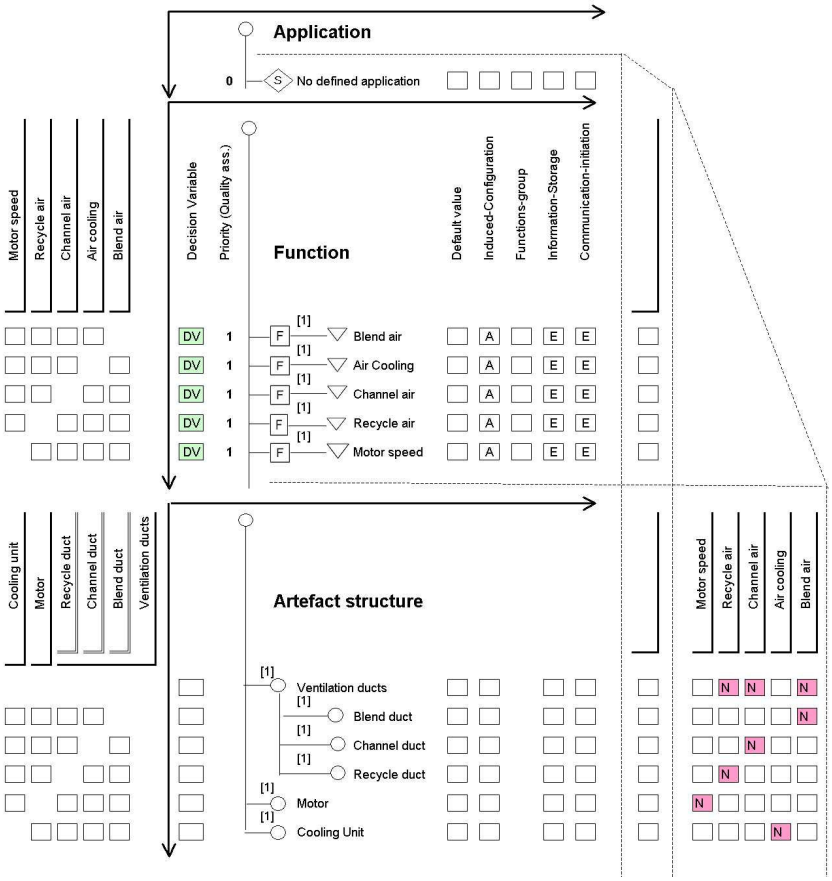



**Figure 118 - HVAC without values**

The model in Figure 118 shows that there are five DVs. and there are no relations between them, nor is there defined an application. This leads to a flat model; no cascading effect and all functions are DVs. When an application is introduced, here identified as the temperature inside the cabin (we assume single zone HVAC), the model changes dramatically (Figure 119). New DVs are introduced (two: *Cabin temperature* and *AutoOff*); these then connect to the previous DVs with some calculations, making the old DVs dependent on the new. Three new functions are introduced but only as internal variables (marked *Dynamic Variable* and hence cannot be changed directly). These have values since all three variables measure temperatures in different areas of the car. The values along with the required cabin temperature are used to calculate values for the five tentative DVs each time. The DV *AutoOff* changes the state of the system. It turns off the automation and thus reactivates the five tentative DVs. So, the two application variables can be said to be *state* variables, as they *switch* the system between two different modes or states. All this can be seen in the model in Figure 119.

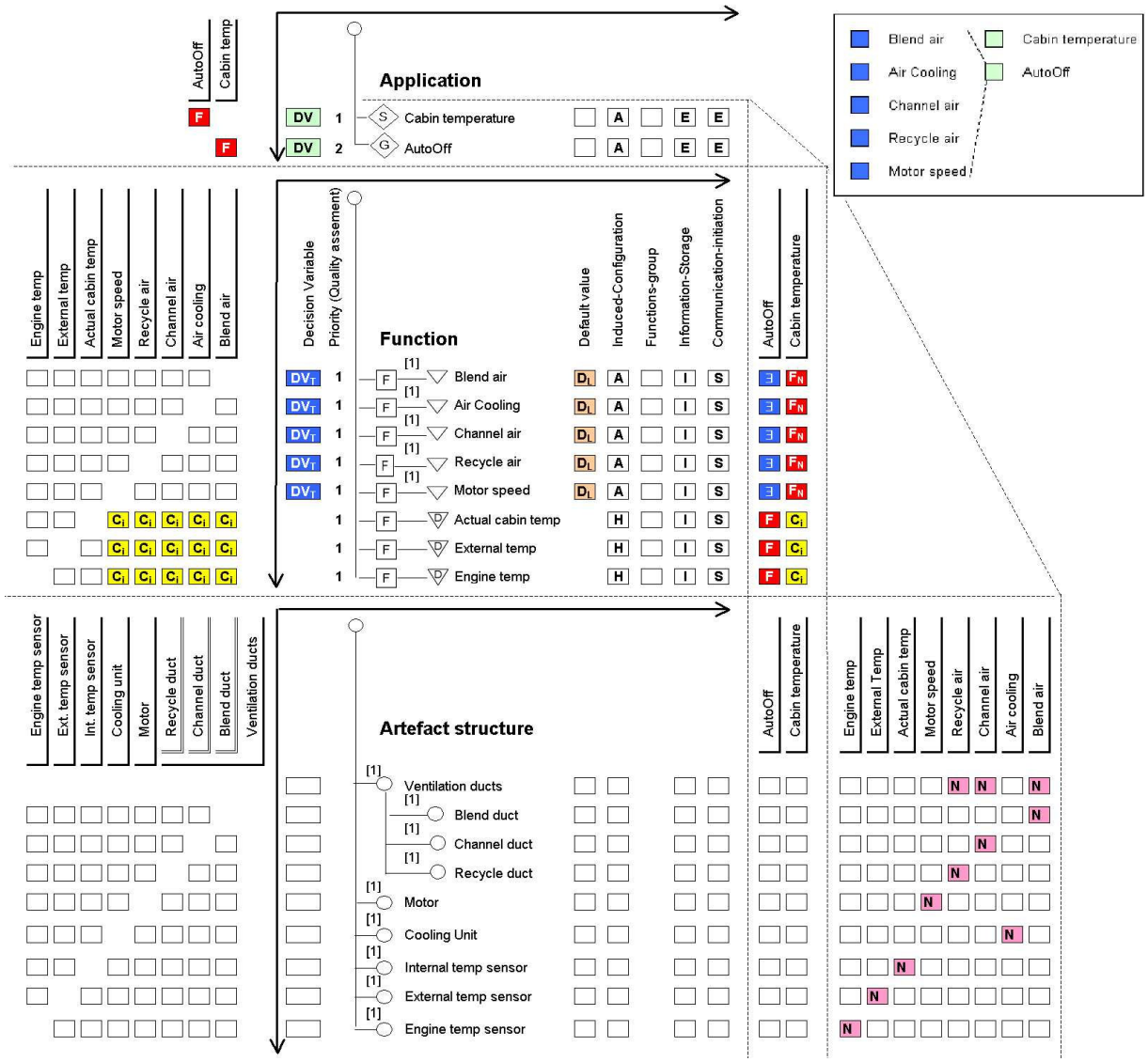



**Figure 119 - Climate control and switch**

Decision variables can be shown explicitly on the top right-hand side, but this is only practical when there are not many DVs. This is helpful to highlight DVs but can be confusing, if there are way too many DVs present.

### 5.4.5 STACKING ENCAPSULATION MODEL

A system is our creation; we choose what we see as a system. Rooted in this, and greatly discussed in system theory, we can decide on what level we draw an EM. One could easily make an EM for whatever system, with whatever boundary. We can even make a layered EM, one for each subsystem, and then one for the whole. The wider one would be a *Stacked EM*. HVAC is actually a stacked EM for a complete system. The purpose of making a stacked EM would be to facilitate visualization of the actual solution space for a specific hardware setup. This could be considered an application model. The main trick in making a stacked EM is to generate it automatically, based on the participating EMs from the subsystems. We return to this later, when we go through an example, but now let us look at the communication model and its intricacies.

## 5.5 THE COMMUNICATION MODEL

The third model is the communication model or *CM*. This is a guideline model to help the encapsulation model define and execute information exchange. The communication model builds on the SBM and feeds the EM. The purpose of the CM is to define the borders of the system, what information is stored where, what information is allowed in the form of an ontology, and how to communicate between EMs. The purpose can be said to be two-faced:

- Adoption of a standard language so things can be matched, including what connects to what.
- Information on what constitutes a working system, what is needed, what is optional etc. An important factor is what emergent properties are to be included in the final system.

The CM is basically ontology, a kind of "Tell & Ask" ontology (see page 100), where we define what information is allowed and how it is connected. The general goal is to move as much information as possible from *Obtain, Receive* and *Provide* to *Present,* as shown in Figure 120 and Figure 121.

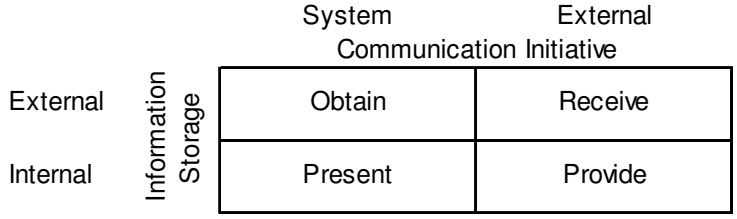



**Figure 120 - Communication as information storage and initiation**

The reason for this is that we want the system to store as much of the information needed for communication in phase 2 and 3, so automation can be achieved. To allow automation, the system also has to have the communication initiative, hence the move described earlier. When information is stored within the system, and the system is given power to initiate communication, much simplification can be achieved. In the *present* field, it is important to remember that two-way communication is going on, both *Inform* and *Request,* as shown in Figure 121. As a consequence, it is necessary to identify all elements that are *inform* and *request* and then pair them so that a specified inform answers a specified request. This requires the language to be predefined and some rules to be made for pairing so that it can be undertaken automatically.

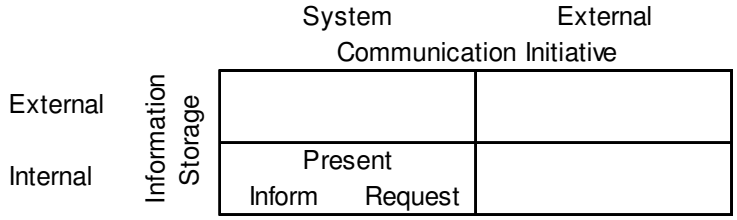



**Figure 121 - Present has both inform and request actions**

Work in the field of artificial intelligence on agents and agent systems (Russell & Norvig 2003) has already suggested a solution to this. It is not our intention to re-invent the wheel here, so we use the theory of intelligent agents (Wooldridge & Jennings 1995) as our communication structure basis (FIPA 2002a) and only deviate where we find it is necessary.

This has several consequences. We obtain a communication framework complete with language, rules of communication, vocabulary for speech actions, behaviour models and connection to ways for implementation. What we do not obtain are naming conventions for the abstraction levels, connection between EMs and CM, and a way to "visualize" the overall CM. Let us start by describing how we use the intelligent agent framework, and then move on to solve the issues for which the framework provides no suggestions.

The foundation for intelligent physical agent (FIPA) standards suggests a communication framework that includes definitions for all the necessary speech actions. Even though FIPA's definition includes 22 actions (FIPA 2002c), we rely mainly on *Inform* and *Request,* and use the other 20 only if necessary. This is done for simplification, and based on the fact that the other 20 are derivatives of the first two. Inform and request actions have to be encoded into the EMs and linked to the inner workings of each. For the rest of the framework, we use it as it comes. For language, we use the FIPA *Semantic Language* (SL) content language. FIPA SL is chosen for several reasons. First, it is complete and can therefore describe any situation. Secondly, it has evolved through decades of research in AI for semantic languages. Thirdly, it is an international standard and thus independent of profit makers (companies). And lastly, it is the author's belief that use of standards should be facilitated wherever possible; we should not spend time "re-inventing" the wheel.

Let us now look at the issues that are not solved by agent theory. The core of these issues is the mapping between the framework and the actual domain – for example: what information is necessary, who makes requests and for what, how are the agents identified that have to exchange information, and finally, how does all this fit into the overall method presented in this thesis.

To achieve this, we believe it necessary to link the higher abstractions of the EM to the CM; this applies both to the function and application levels in the EM. Since the description of these levels can be quite defuse and without known building blocks, it would be hard to introduce automation into the process. By process, we refer to the general process of installing the complete system.

Therefore, we suggest using a naming convention to describe the elements in the EM. Elements at the artefact level do not need a stringent naming convention, because their use should only be internal in the EM, and all the influences the artefacts have on the overall system should be mapped through the functions (or applications). The functions level is the core level in terms of what needs to be communicated between EMs through the CM. Hence, we suggest a stringent way of naming the functions. We adhere to the verb-noun method of describing function. For the basic functions, we choose to use *Functional Basis* (Hirtz, Stone, McAdams, Szykman, & Wood 2002), as it is complete, tested and should cover all the aspects needed. We also recognize that describing functionality with basic only functions is tedious and very hard to describe to other people; therefore, it is necessary to group the functions. Such group names should still be verb-noun, but more closely related to the domain that is being described. Group functions should coincide with the SBM and be relatable to the breakdown suggested there.

Predefined naming conventions for application descriptions do not seem to exist, at least not that we have found. There have been some attempts to suggest such vocabulary. Hubka talked about *effects* but did not compile such a set. The most relevant suggestion found is the proposal for *organ units* made by (Jensen 1999), who did not suggest a set either. We recognize that it probably would be beneficial to have such a vocabulary. It would be wise to develop such a tool, a sort of *Effect Basis*, but for now, we suggest using the verb-noun approach for functions and intentional functions. The same could apply for the applications, since a tight relation exists between intentional functions and applications, both in the form of services and requirements.

Returning to inform and request, let us show one way to visualize such relations.

### 5.5.1 VISUALIZING INFORMATION FLOW

Borrowing from SysML (OMG 2007), the parametric diagram is extremely suitable for visualizing inform-request relations. It has a predefined lingua, order of flow and input to output through black boxes, i.e. the diagram focuses on the inputs and outputs and treats the rest as a black box. A parametric diagram for a pump system can be seen in Figure 122.

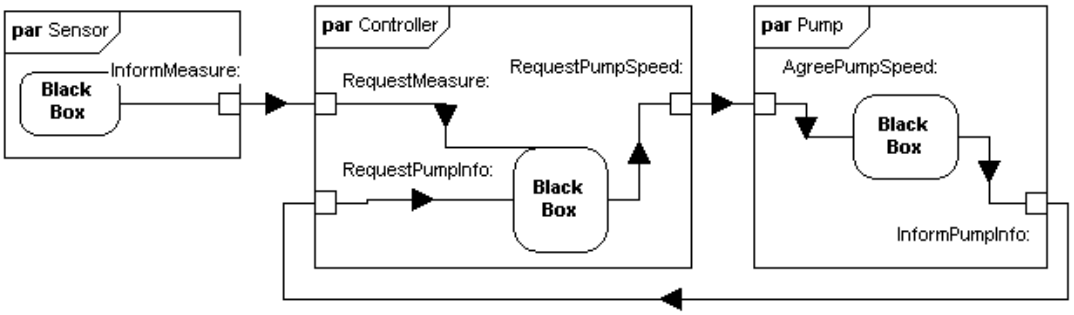



**Figure 122 – Part of a parametric diagram for a water system.**

Naming the variables with speech actions from FIPA allows variables in sub-systems to be matched. Of course, a parametric diagram can be drawn without such names and would then point to the information flow, and thus suggest the required names for each variable.

### 5.5.2 THE CM

The communication model is not a model in the strictest sense. It is a guideline for finding and showing the information flow needed to facilitate communication between sub-systems. The CM consists of:

- Verb-noun naming convention, with use speech actions to limit verbs, and domain knowledge to limit nouns.
- Elements in application level that are compiled to form a domain basis, a *Services Basis*.
- Elements at the function level that:
    - for intentional functions should be compiled to form organs, a function with purpose, an *Effect Basis*.
    - for basic functions, should use the predefined *Functional Basis*.

- Elements in artefact level that are described with domain language.
- Classifications of communications according to the four types of behaviours: identification of HIC and AIC, which then lead to definition of DVs, Tentative DVs and Internal DVs.

When conceiving the CM, we should pay attention to the following points:

- What information is to be matched for in and out sharing?
- Use both the archetypes to generate the *pair* boxes, and the SBM and EM for each artefact to find the inputs and outputs.
- Do not draw internal workings as this only complicates matters.
- What the system can expect to find. Construct a list of questions based on SBM. This could be done with a master EM (without matrixes) or ontology.
- Is there a different CM for each phase? Or is the CM broken into phases, too? A single CM is probably preferable, but each case has to be examined and evaluated.

This concludes our presentation of the modelling technique. The next chapter reports on four tests that were made of the method.

# Chapter 6

# TESTING THE MODEL

This chapter is about the testing of the models to prove their worth. Since this study has been explorative in nature, and in fact several mini-tests have been made underway, the four tests described in the following section are the main test supporting this thesis. The tests were also explorative in nature and had few participants. They were relatively unstructured tests and relied a lot on the competences of the participants. Their main purpose was to provide iterations meant to improve the method but also to partially test their worth.

The four tests described here involve functional descriptions, making PVM centrepiece, modelling relations with a lot of data, and finally running the whole set of models on a product system. The tests are described in that order.

## 6.1 TEST #1 – FUNCTIONAL DESCRIPTIONS

This first test's purpose was to make a functional description of a product in a stringent way with *Functional Basis,* and to see if it could be incorporated directly into the method. This first test is rooted in the notion that functional modelling is difficult, and if previous work could be directly integrated, this would probably be highly beneficial. After extensive literature search on functional modelling, Functional Basis was chosen due to its completeness and apparent qualities.

### 6.1.1 SETUP FOR TEST #1

The test was conducted at the case company site in a single one-day workshop with participation by a domain expert at the case company who is responsible for software development in a supporting PC tool. The workshop was conducted in the summer of 2006.

Some time in advance, a document describing the Functional Basis (FB) method was sent to the participant and a product was selected beforehand as the modelling object. Documentation on the current parameters used in the artefact was known and had been studied by both the expert and the researcher. The aim was to group or structure the existing parameters according to the FB method and its verb-noun formalism:

- An electronically controlled centrifugal pump, a so-called E-pump, named CRE at the case company.
- Artefact variance in the order of 150.000, done by combining physical modules. The variance created with software is not known, i.e. how many different setups can be made.
- Embedded software has 160 parameters that can control both setup and behaviour.

### 6.1.2 CONCLUSION ON TEST #1

It became quite quickly clear that the FB would produce a detail level that was completely unnecessary for the purpose of the method. The labour for constructing the functional description of the artefact in relation to the expected benefits was enormous. As the FB is made to design in detail, it is very fine-grained. It has no predefined groupings. Also, FB is not lined up to deal with software variables. Before the day was over, the test was abandoned. The participants realized that during the day of the workshop they would only be able to produce a fraction of the functional model and that way too many details would be involved.

### 6.1.3 COMMENTS ON TEST #1

Even though the test was abandoned, it was by no means a failure. Several things can be learned from it.

- Detailed functional descriptions that are appropriate for designing artefacts from a clean slate are not so useful when it comes to describing functionality in software. Or better stated, Functional Basis vocabulary is way too detailed and therefore requires too much grouping to be directly useful in describing higher-level functionality.
- This leads to the following proposition: A higher-level functional vocabulary is needed and may have to be developed.
- Thinking solely about functions is EXTREMLY difficult. We have a tendency to frequently "fall down" to either the artefact itself, or in this case, to the actual software implementation of the parameters. So the need for both a complete functional language and a method for generating the models is quite clear.
- Relating functions in a formal way is even harder than just stating them. Even though a lot of literature was read on functional modelling and approaches, no literature was found that focused on functional relations. Most authors completely avoid the subject, while a few give it superficial treatment. As the core premise of this thesis is to create complete mapping, from application (the customer need) down to the artefact through functional description, it is extremely necessary to have a clear view of these functional relations.
- Dealing with this functional modelling of software-controlled artefact variance would probably have been enough material for a PhD thesis.

## 6.2 TEST #2 – POPULATING THE PVM

The purpose of the next test was to try out population of the PVM and see if relations could be added. The focus here is to generate the centrepiece in the PVM with appropriate decomposition and look at relations between parameters. The last part of the test was to see if standard configuration software could be populated with the model.

### 6.2.1 SETUP FOR TEST #2

The setup for the second test was as a special course at DTU. Engineering students with focus on product design were given a product from the case company with complete documentation, along with access to a domain expert. The students were at Bachelor level. The course ran during spring 2007 and the researchers supplied both literature and guidance during the period of the course. The students were given overall guidelines for what to achieve, but the researchers did not give them detailed descriptions of methods of decomposition and relationships. This was not done out of malice; it was simply because that work was not completed at that time. So this test was highly explorative, and the researchers conducted many discussions with the students on almost all aspects of the project.

The product used in this test was a fresh water booster system named HYDRO 2000E. The pumps in this system are not electronically controlled, meaning that they run on or off. The system has the following attributes:

- Only the controller for booster systems (a pressure increasing system) has an internal computer.
- It is an older product so the processing power is limited and also the number of variables.
- The system contained 155 variables.

The students presented their work in a report and an oral presentation of their proceedings. They received 15 ECTS points each for their work.

### 6.2.2 CONCLUSION ON TEST #2

This test was a very iterative and explorative process. The students were placed in situations where they were in deep water and had to rely on all their combined knowledge, experience and common sense to work through the project. This points to a higher complexity in the method than the researchers had foreseen and made it apparent that more training was required. The students grasped the method, even though it required some work for them to become acquainted with it. Four main aspects can be reported from the process: *Understanding the Artefact*, *Decomposing*, *Relations* and *Software Support*. Let us look at them in this order.

- The first aspect involves artefact understanding or the gaining of domain knowledge. It was hard for the students to relate to the artefact, as they had no domain knowledge and had to ask or read about everything. This is to highlight the fact that making a model of artefacts and their inner working is of course extremely dependant on domain knowledge. On the other hand, the students are not as "locked-in" in their ways as many domain experts are and are thus able to move "outside" of the box. At the time of the test, this was not considered so important, but it turns out that it is very important.
- Decompositions are the second aspect to highlight. By decompositions, we mean both groupings and the breakdown of artefacts, functions and applications into sub-units. It turns out that achieving decomposition is difficult, especially when the modellers do not have either domain knowledge or specific guidelines for the decomposition. In this case, the prevailing structure in the documentation tended to dominate the outcome. As expected, functional descriptions were hard to make and

a lot of discussion went into how to achieve them. In general, we can say that arriving at the final decomposition was a painful trial-and-error process that went through several iterations.

- The third aspect is relating elements in the model. With no predefined relations, the process of relating was very time-consuming. The students had some very good thoughts on relating parameters. The process was iterative. First, the centrepiece in the PVM was made, refined and grouped. Relations between application and artefact were made first as most parameters were on the artefact level. Later, analysing the application-artefact matrix populated the function-artefact matrix by introducing functions in between. The last step was structuring and grouping the relations and finally improving readability of the method. The readability was connected to types of relations, e.g. drop-down values versus binaries (true / false). Rephrasing or regrouping often produced a more readable matrix.
- The final aspect to report is software support. As we have pointed out throughout this thesis, it is necessary to support the modelling technique with IT. The manual labour involved is enormous, even if we use such tools as Excel. The students made a prototype in standard configuration software (ARRAY technology) to show how decomposition into three abstraction layers could be achieved and how the cascading effect would work. As Array is a matrix-based system and relies on tree structures (like so much other software) to represent decomposition, it had no problem showing the centrepiece and its three abstraction levels. The relations were not shown in matrixes, but a programmer at Array assured us that adding such visualization would not require a tremendous amount of work. The students and researchers did not test this statement but admit that it probably holds, based on the nature of both the software and the method.

Let us close the conclusions on test two with a general comment on the whole. The completed model shows that the product was very "flat" in structure; it had very little cascading effect. The students suggested several applications and linked them through functions to the artefact and its parameters. When the model was presented to the domain experts, the flat structure came as no surprise to them. But as this was an old product, it was challenging to engage the domain experts in discussing the usefulness of the method. It is thus important in later tests to have an artefact that is "closer" to the everyday reality of the domain experts, if we can expect useful feedback from them.

### 6.2.3 COMMENTS ON TEST #2

From a research point of view, a single lesson stands out in this test: The fact that if we are not relatively specific about what is to be achieved and how to achieve it, progress will be really slow. Here, it is the lack of decomposition techniques and predefined relations that really slow the work down. If a method like the one suggested in this thesis is to have any chance of being accepted, we have to supply guidelines for decompositions and predefine a set of relations along with guidelines for using them.

## 6.3 TEST #3 – SIZE AND RELATING ELEMENTS

After observing test two and its conclusions, it became apparent that more testing was needed. The work on defining relations had progressed at this point, so a set of relations and guidelines for using them was ready. The purpose of this test was therefore to check relations with a large number of parameters. The questions of interest in this test were: How big does it get? How much are the relations matrixes filled in? How can matrixes be trimmed? And is it at all possible to obtain a single model overview of a really big product (in terms of parameters). A side aspect was: How difficult is it to make the model without specific IT support? This last aspect is rooted in problems observed in the second test.

### 6.3.1 SETUP FOR TEST #3

The setup for the third test was as a special course at DTU. Engineering students with focus on product design were given a product from the case company with complete documentation, access to a configuration tool in a PC and to a domain expert. The students were at M.Sc. level. The course ran during autumn 2007, and the researchers supplied both literature and guidance during the period of the course. The students were given overall guidelines for what to achieve, along with somewhat detailed descriptions of methods for decompositions and relationships. This test was not as explorative as the second one, but still relatively explorative. The researchers conducted many discussions with the students during the project. Most discussions were on relations and methods for trimming the matrixes.

The product used in this test was a controller for submerged pumps, called CU401. It has the following attributes.

- It is a new product with 1368 variables, which were reduced in the test to 599 by ignoring cardinality.
- Product information was gained from documentation; reverse engineering of a configuration tool (PC Tool Modular Controls) and dialogue with domain experts.

The students presented their work as a guideline report, a poster (full blown model), and an oral presentation of their proceedings. They received 7.5 ECTS points each for their work.

### 6.3.2 CONCLUSION ON TEST #3

This test's main purpose was to populate the model with a lot of data and observe what happens in terms of work needed to make the model, the overview it gives, and how it can be used to analyse the artefact. Three main aspects can be reported: the *size of the models*, the *relations* in both matrixes and decomposition, and finally, the *analysis* of the model with an eye to suggesting improvements. Let us look at these aspects in that order.

The first aspect is the size of the models. By size, we mean both the decomposed tree with parameters and values, and the relations shown in six matrixes. The size issue is also about how to reduce size. The third element that size relates to is IT support. Let

us look at these three aspects of size: *Model size* in number of relationship boxes, *reducing size,* and finally *IT support*, in that order.

- The PVM contained 599 variables instead of the 1368 variables that were found in the data source. The reduction is due to ignoring cardinality. Distribution of relations between the matrixes is shown in Table 36. As these numbers show, the size of the model is almost solely in the internal Function matrix, which is the biggest of the matrices. A visualization of the sizes is shown in Figure 123 and Figure 124.

**Table 36 - Matrixes and their size**

| Matrix | Variables | | Matrixes with variables | and values (Figure 123) | Collapsed (Figure 124) |
|---|---|---|---|---|---|
| A-A | 2 | **0,3%** | 2 × 2 | 2 × 2 | 2 × 2 |
| A-F | | | 2 × 581 | 2 × 985 | 2 × 985[2] |
| A-S | | | 2 × 16 | 2 × 41 | 2 × 41[2] |
| F-F | 581 | **97%** | 581 × 581 | 985 × 985 | 985 × 154 |
| F-S | | | 581 × 16 | 985 × 41 | 148 × 41 |
| S-S | 16 | **2,7%** | 16 × 16 | 41 × 41 | 41 × 18 |
| Model size[3] | 599 | | 348.311 | 1.014.347 | 194.869 |

[2]These matrixes are not collapsed. [3]Size measured in number of relation boxes in the matrixes

- Trimming the matrixes and making them as small as possible is a great task. There are some methods for reducing the size of the matrix. The most obvious one is to collapse empty columns in the internal matrixes. The size of the F-F matrix can be reduced six times to a total of 154 columns with relation boxes. This equals 3½ A0-sized pages. Another method is to remove the space between relation boxes. This makes everything more cramped but also reduces the size drastically. This is a fine way to produce the "colour palette" needed for visual scoping. The internal matrixes are trimmed and the results can be seen in Figure 124 (in comparison to Figure 123). It is also possible to collapse the mapping matrixes.

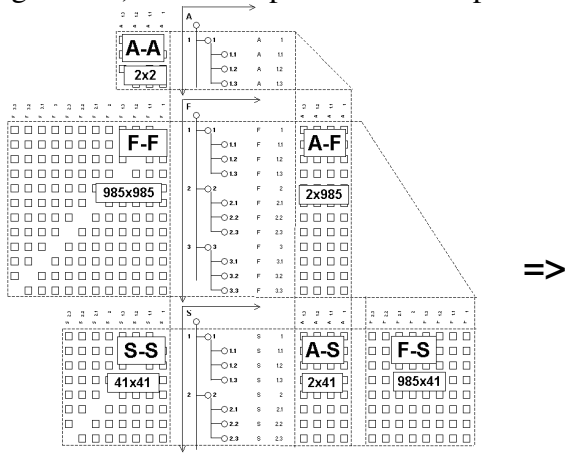



**Figure 123 – Full-size matrixes**

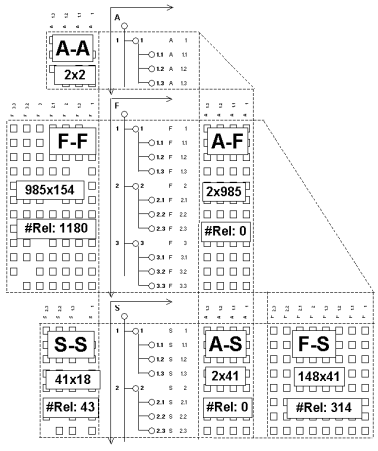



**Figure 124 - Collapsed matrixes**

- The manual labour that went into making the model was enormous. The tool used was Excel 2003, and its limits were reached, as each sheet can only hold 256 columns. This has been improved in Excel 2007, where 16.384 columns can be handled pr. sheet. Because of this limitation, as highlighted in Table 38, the number of Excel sheets used is a fine indication of size. Another thing is that simple tasks, like moving columns, are very tedious because of lack of IT support. So all

manipulation of the matrixes is really tedious, and it can easily be stated that if the method is to have a chance, IT support has to be increased drastically.

The second aspect to be reported is the relationships. Here, we mean both the matrixes but also the decomposition of the PVM tree. In the tree, the important question to answer is when to show values, as this causes the matrixes to "blow-out" in size. A third problem involves the filling in of the matrixes, i.e. how many relationship boxes are populated with relations. So, let us look at these three problem areas in this order: *tree decomposition*, *relations in the matrixes* and their sizes, and finally, the *fill-in percentage* of the matrixes.

- First is the tree decomposition. As the tree directly influences the size of the relational matrixes, it is very important to be as compact as possible in decomposing it. Each parameter was evaluated; if parameters were binaries in nature, they were phrased in a true / false statement to only use one line. Where values are involved, serious thought was given to whether values were to be shown. Even though grouping is helpful in structuring, we have to be constantly aware that unnecessary grouping causes the matrixes to expand. Decomposition guidelines could have been used but where missing. To apply the method properly, further guidelines are needed, as the task is too difficult without them. All the parameters were evaluated for AIC / HIC and default values. Default values are kind of misleading, as defaults are achieved by file download and not predefined values. The AIC / HIC split was about 95%/5%, which is surprisingly low on the HIC. This is approximate, as domain experts do not always agree on where a parameter lies. This highlights the fact that there is a "soft limit" between AIC and HIC, which is highly dependant on system view, what is inside and outside the system.

- All relations found in the supplied documentation and gained from interviewing the domain experts were put into the matrixes. In contrast to earlier tests, the predefined relations helped a lot in constructing matrixes. Before the test, the researchers supplied twelve defined relation types. Eight of them were used in the matrixes. The types are "exist", "equals", "bigger than", "smaller than", "true", "false", "default" and "lowest". Table 37 gives an overview of the number of relations found. No relations were found for the two variables in the A-F matrix, resulting in an empty matrix.

**Table 37 – Overview of the relations in three of the matrixes**

| | F-F matrix | S-S matrix | F-S matrix | A-S matrix |
|---|---|---|---|---|
| Exists | 1071 (90,8 %) | 40 (93,0 %) | 312 (99,4 %) | 1 (50%) |
| Default | 1 (≈ 0 %) | 0 | 0 | 0 |
| True | 0 | 1 (2,3 %) | 1 (0,3 %) | 1 (50%) |
| False | 0 | 2 (4,7 %) | 1 (0,3 %) | 0 |
| Other | 108 (9,2 %) | 0 | 0 | 0 |
| Total | 1180 | 43 | 314 | 2 |

The high number of "exist" relations reduces the degree of cascade effect, since user-prompted choices are required for almost every variable. The external Function-Structure matrix is the larger of the two external matrixes. If the model were not reduced in regard to identical variables (cardinality), the matrix would have 689 relations, as opposed to the 314 relations that are present in the matrix.

- The main purpose of the suggested method is to give an overview of relations between (and within) abstraction levels in order to evaluate the cascading effect of parameters. In relation to the last problem of relationships, some statistics for the relational matrixes is in order, as shown in Table 38. It can be seen that the matrixes, even collapsed, are quite "empty".

**Table 38 - Relationship fill-in in matrixes**

| | F-F ignoring cardinality | F-F with cardinality | S-S | F-S | A-S |
|---|---|---|---|---|---|
| | *Collapsed* | *Collapsed (estimate)* | *Collapsed* | *Collapsed* | *Full* |
| Variables | 581 | 1332 | 16 | | |
| # Values | 985 | 2360 | 41 | | |
| Dimension | 985 × 154 | 2360 × 250 | 41 × 18 | 148 × 41 | 2 × 41 |
| # fields | 151.690 | 590.000 | 738 | 6.068 | 82 |
| # relations | 1180 | 2300 | 43 | 314 | 2 |
| Fill-in % | 0,78% | 0,39% | 5,83 % | 5,17% | 2,44% |

These numbers give an indication that can aid in redesigning the artifact and its control mechanism. We return to this in the analysis part. A more detailed view of the size and fill-in percentage for the function-function (F-F) matrix shows that the model becomes really large when printed as listed in Table 39. The F-F matrix, even in its most collapsed form, is a BIG printout. The full size with cardinality would fill a small house. A printout of this size will not provide the user with any useful overview of the product. The 3,5 A0 printout is at the outer most limits of practicality. In this regard, more efforts are needed to suggest further modes of dealing with size. The most obvious is to use IT support and allow multi-layer viewing of the model, maybe in connection with allowing the decision to view "branch-level".

**Table 39 - Relationship fill-in in F-F matrix**

| | F-F ignoring cardinality | | F-F with cardinality | |
|---|---|---|---|---|
| | *Full* | *Collapsed* | *Full (estimate)* | *Collapsed (estimate)* |
| Variables | 581 | 581 | 1332 | 1332 |
| # Values | 985 | 985 | 2360 | 2360 |
| Dimension | 985 × 985 | 985 × 154 | 2360× 2360 | 2360 × 250 |
| # fields | 970.225 | 151.690 | 5.569.600 | 590.000 |
| # relations | 1180 | 1180 | 2300 | 2300 |
| Fill-in % | 0,12% | 0,78% | 0,04% | 0,39% |
| # Excel sheets | 9½ | 1½ | 22 | 2,4 |
| Printed size[1] | 22 A0 | 3½ A0 | 126 A0 | 13½ A0 |

[1] Printed at 40% resolution, column width/height = 20 pixels, Tahoma font size 10

The last aspect to report is the analysis of the model. The purpose of the method is to work both as an analysing tool but also as a design tool. The work done in this test is

highly focused on the analysis of an existing artefact, but with improvements in mind. So, it is an AS-IS description of the artefact. The analysing capabilities of the model can be grouped into three groups: *detail tracking* where cascading effects are checked, *overall visual* with its palette view, and finally, *hardware present* mode to check for consequences of removing modules. Let us look at these in turn.

The first is the detail tracking through the model. This aspect of the model is the basis for all other activities. Relating parameters through the internal mapping matrixes gives a map of the artefact.

- Tracking through the matrixes the consequences of selecting a specific value of a specific parameter. With a top-down reading of the model, we should be able to deduce the cascading effect by seeing how many parameters are set with a choice of a single one at application level. Current structure has almost no applications, but inspection shows that the functionality and the intended applications actually lend themselves to definition of several services (applications) that would / could generate a massive (big) cascading effect. We are talking about serious reduction of DVs. The actual number is not available, as the test was not taken that far. Again, the product shows a relatively "flat" structure. Most parameters are functions (understandably). They are not so coupled internally, and linkage to artefact is often "unclear".

Second is the visualization of the complexity. With all the relations in place, we can start to "visualize" the complexity of the product. This can be done in two ways: *pattern observations* – what is there, and equally, what is not there; and then *colour coding* the relations and looking at a palette.

- What do relations look like? Are there patterns? What is there and what is not? In this case, just the placing of relations points to a state. The A-F matrix is empty! That means that there is no application or service thinking in this structure, at least not in regard to the researchers' view of such structure. Functions are not tied together, and those that do have relations have only "exists", meaning that the user has to take action on them. If the internal matrixes show a kind of diagonal pattern, a cascading pattern, the structure is most likely uncoupled.
- The use of colours in each relationship type creates a palette in the matrixes. With an intelligent mapping of colours and relations, we should be able to generate a "picture" that is easy to scope! For example, in this test, the colours red and green are "good" colours as they set parameters to a specific value. In these terms, the colour blue is bad, as it requires user input. This could therefore be better. So a single blue colour would point to a really flat structure with little cascading effect. The main attribute of such a palette method is that it allows a zoom-out to generate an overview of relations with colours without being able to read individual texts. A complicated structure can be evaluated with such a picture, based on hundreds or even thousands of parameters and values.

The hardware present is the last aspect to report in analysing the capacity of the model. Even though the model technique is made for top-down reading, a possibility exists in finding out the effects of hardware on the functions and services available.

- Backtracking through the model is hard to do manually, but as the model is made for use with configuration engines that can work through the rule set in any order, or at least backtrack through the rules, it is very probable that hardware dependencies can be established. This aspect should be kept in mind when

constructing the relations, as this could be very beneficial in an autonomous system setup and not least in the operation when failure occurs. It can most likely be stated straight away that this is first possible when IT support has been made available for the models.

A concluding remark for test number three is therefore: The need for IT support becomes VERY apparent in this test, and the modelling technique will first be fully usable when such support is available.

### 6.3.3 COMMENTS ON TEST #3

The first thing to be noticed here is that the students were much more at home in this test in contrast to the previous one. This is most likely due to their combined experience, as one of them was in the previous group. An obvious knowledge transfer did take place, and the students thus "bypassed" many of the pitfalls hit by the other students. Another thing is that using actual implementation is very good, meaning a software tool, and then reverse engineer it to capture its logic.

The researchers' remaining comments on this test can be summarized in three groups: *the model* and its characteristics, *the data* populating the model, and finally the process and *the IT support* needed.

- The first thing that strikes us is the shear size of the whole model. If a product has several hundred (or even thousands) of parameters, the model becomes very large. On the other hand, it is surprisingly easy to track through the model and see the consequences of a single choice. This is very fortunate as it supports the whole premise of this thesis. The other aspect is the palette effect, the use of colours to enhance our visual capacity for analysing the model. It makes a great difference to have colours, because they help us to quickly "get a feeling" for the model, its complexity and possibilities for improvements.
- Once the model has been populated with data, several things become apparent. The first thing is that the complexity of rules is not high; it is always the same types, and there is not a single relation in the test product that the students were not able to put into the model. This is very good. Maybe it shows that humans intuitively try to simplify relations for easier understanding. The model showed that the product has a "flat" structure, meaning that there is very little cascading effect in it, and most parameters require user inputs. This is of course solved in practice with downloadable files. Another aspect here is that the HIC / AIC ratio is much lower than expected. Only around 5% of the parameters is hardware-related. This is a little surprising, as the researchers had expected this to be much higher. It would be very interesting to research why this is so. The last aspect that stands out in the model work is the fill-in ratios of the relation matrixes. Fill-in ratios are from low (about 6%) in the mapping matrixes to very low (<1%) in the internal function matrix (see Table 38). So the question arises: Is this method appropriate? Is it too detailed? Is there need for a wise grouping mechanism to reduce tree size and increase fill-in of the relation matrixes at the cost of losing details in relations between specific parameters? These questions provide material for yet another test or two.
- Another highlight from this test is the process and IT support needed. All manipulation is very time consuming; just minor changes to the model can easily

take hours. The size of the model also causes problems. There is no dynamic overview with different detail levels where we can "change" between levels at a click. This is a feature that is most likely needed in order to make the model and the modelling method attractive for any practical use. To spend most of the time "fighting" with Excel is probably not very productive.

As a quick summary, the model works but there are "issues" that have to be fixed before a major application can be suggested. Future steps in improving the model and method would include conceptualizing "breakdown" practices to allow for dynamic over viewing. This should include some grouping mechanisms, suggestions on how to use branches and levels in the PVM tree to facilitate decomposition, and of course, finally, implementation into a software tool.

## 6.4 TEST #4 – MODELLING A SYSTEM

The last test was a workshop at the case company. The purpose was to run the whole set of models through with some practitioners and get their response to the method. It was left open what products were to be analysed, and at the start of the workshop, the group chose one system.

### 6.4.1 SETUP FOR TEST #4

The test took place at the case company. The company had allotted resources and a location for the workshop, and there were no interruptions during the whole two-day workshop. The test was run in late spring 2008.

The domain experts present were members of the R&D team from the case company and responsible for the embedded software in the product assortment and supporting tools. As much of the variance of the products is made with black boxes and software, these domain experts have thorough understanding of the products, both regarding function and structure. One of the domain experts was the same as in first test. All domain experts have been involved in the project and are not new to the concepts intended for testing.

The researcher opened the workshop with a presentation of the concepts. The rationale for the why's was presented and expected benefits discussed. A document describing the concepts had been distributed the week before to all participants. Throughout the workshop, the researcher was actively involved in modelling the selected system. The researcher tried to convey the concepts along the way and answered all questions that arose. The researcher also engaged in discussions on rationality, validity and usefulness of the suggested concepts. One-fifth of the workshop duration was used on presentation of concepts and four-fifths on modelling the selected system. The system selected for analysis was a typical product for the case company, a product that has all the major elements such a system exhibits and complexity above middle.

The plan for the workshop was to go through the making of archetypes, FSM, SBM, CM and EMs for a complete system. In retrospect, this was an ambitious goal for a two-day workshop. The participants did not get to make all the models, as expected.

Most of the time was spend on archetypes, FSM and SBM. There was only very little discussion on the EM and none at all on the CM. The highlights from the test are discussed in the following.

### 6.4.2 CONCLUSION ON TEST #4

There are three main aspects to report from the workshop: the *effectiveness* of the models, *the process* of making the models, and the underlying *knowledge base* of the participants. Let us go through these, one at the time.

The first main thing to report is the effectiveness of the modelling method. The participants understood the main message conveyed about the purpose of the method. They saw the point in talking about system boundaries, decision variables that need user inputs / actions, finding out whether parameters have to do with hardware setup or application (HIC / AIC), and finally linking parameters together to look for cascading effects. There was a general acceptance to the validity of the method, though some concerns were raised. These can be grouped into three main issues: *Archetypes and FSM* issues, *functions and interface* issues, and finally, *rule complexity* issues in EMs.

- In making the archetypes, lively discussion arose. Questions like: "What is a good archetype" and "what are the characteristics of a good archetype" triggered talk on how to create archetypes. No conclusion was reached in the discussion other that the archetype should be in domain language and representative of the artefacts made in the company. One participant pointed out that in Software Engineering (SE), the SuD (System under development) is not drawn into the system picture. This is of course very valid if clean-slate design is being performed. If the work is re-engineering or analysis of exiting artefacts, then this is hard. After the group agreed on archetypes, a lot of time was spent making functional streams (FS) for the system. The trick was to make the FS abstract enough so as not to introduce far too many elements. After much discussion, the group landed a set of very simple FS that appears to represent the system very well. The next challenge was to overlay the AT and FS. This was not so difficult. There was a quick consensus on how to do this, and a SBM was drawn with relative ease. One participant pointed out that the FS have many similarities with UML's Use Cases. Both are aimed at generating an understanding of the overall system. The researcher constituted that Use Cases could easily be applied instead of FS to generate the SBM, which is the conglomerated result of the overall system picture. During the whole process, it is important to be aware of the complexity of SBM as its usefulness is probably drastically decreased, if it has too many elements (AT and FS).
- The next issue is the functions and their interfaces. So, as the main purpose is to reduce the number of decision variables, which can be said to be the human-system interfaces, the same holds for interfaces between functions. Fewer connections between functions probably means simpler handling, documentation and implementation of system. To deduce the "right" functions at the "right" abstraction level and the "right" connection is not easy. By interfaces is meant both internal interfaces between function (within each archetype) and between functions in different archetypes. The participants pointed out several ways to achieve this. One is to "pull" function (service) out of the system and see what interfaces and / or relations are then required! This should lead to some sort of "stereotypes" (that have

the same interfaces). Another would be to ask: What is needed for functions to work? A good guideline to adhere to is one used when making Use Cases: Services are not a process (that is outside) but a high-level function we can start and stop. This is rooted in the notion that good Use Cases are not processes. To deal with communication, focus should be on simplification of interfaces! As mentioned earlier, the workshop did not spend time on making a CM so the discussion of communication was very limited. One note though from one participant is that for a successful description of meaning, it is probably necessary to introduce a general standard, not a monolithic solution, and the standard should extend across companies. This is a very relevant point for later implementations, but we have not arrived there yet.

- The last issue is the rule capabilities of the method. When constructing the Encapsulation Model (EM), participants ran into problems trying to express some rules they wanted. There where two types pointed out. First was the conditional statement (IF-THEN), where the same parameter is used in the condition and the statement. The other was if there were multiple conditional statements that needed to be satisfied.

The second thing to report from the workshop is issues related to the process of making the models. These issues can be grouped into three issues: the *general approach* as in top-down or bottom-up, the *focus* of the process, and finally the *purpose* of the model making.

- The first issue here is the general approach to the model making, the classical top-down versus bottom-up. Many of the lively discussions that took place during the workshop began as a debate between top-down and bottom-up approaches to gaining completeness. The researcher thought the method to be only top-down, but that only holds for the SBM part. In construction of the EMs, bottom-up is actually used. So, the overall method could maybe be called middle-in. This caused some confusion and hence the lively discussions.

- The second issue is the focus of the process. It turned out that keeping focus was actually quite difficult. It was very easy to stray in all directions. It is necessary to be aware of this and keep strict focus. There are several things to remember here. Models are restricted and constructed by the observer, meaning that we all have "different glasses" when constructing abstractions of the world in form of models. The modelling method is to introduce "common glasses", a way to force a formalized way of modelling. This of course is not easy. So, practical advice: Do not get sidetracked; focus on DV, HIC and AIC; do not get caught in operations talk, or try to describe complex logic – at least not in the first iterations. Use the black box thinking, and for example the parametric diagrams. The whole process requires good management and control in order not to get sidetracked! If there is redundancy in functions, focus on core functionality (as suggested by (Pimmler & Eppinger 1994)). Along with the focus problem, this method also has the problem of many other modelling methods: When to stop digging? When is the model detailed enough? What is to be suppressed and what is necessary? Discussion on these wonderings is taken up in the discussion, chapter 8.3.

- Last of the issues related to the process is that the main purpose of the method should be kept in mind. If the method has a single goal, it would be to reduce the number of DVs for the overall system. Again, it can be hard to keep eyes on the

goal. The method would benefit greatly with some sort of quality criteria to aid in showing the way to a "correct" solution, if such a thing exists. A rule of thumb could be to say: If elements do not contribute to DVs and thus their reduction, they should probably not be included.

Last of the main things to report is the worldview or knowledge base of the participants. It turns out that many of the disagreements, and hence discussions, were rooted in different worldviews. It is the researcher's belief that this is tied to the disciplines of Software Engineering and Engineering Design. This is a little unfounded at the moment, but some points can highlight why the researcher thinks this. Design of artefacts is very often redesign or minor changes to existing artefacts. Studies have shown that up to nine-tenths are redesigns. Therefore, engineering design is used to work with what is and where to go, sometimes stated as AS-IS and TO-BE situations. Even though reuse is also a goal in Software Engineering, it is harder. We humans are better at dealing with artefacts than mind maps like software. Even though design is very experience-based practice, SE is much less used to AS-IS work than ED. Another thing is that ED often deduces a TO-BE state from AS-IS, using the current situation as platform, while "good" software practice dictates a clean-slate approach with focus on the actual requirements (as in Use Cases). Rooted in this, several lively discussions on the validity of bottom-up and analysis of existing systems combined with top-down system description were conducted. This is a core issue! It points to a possible implementation problem that has nothing to do with the effectiveness of the suggested method but with the knowledge base of the participants (or model users). It is therefore very important to check this aspect further and see what it does. Or, it is at least necessary to be aware of this when trying to use methods rooted in Engineering Design praxis in Software Engineering.

### 6.4.3 COMMENTS ON TEST #4

The researcher's highlights from the workshop can be stated in three main issues: *preconceptions,* as in what is underlying and incorporated, *acceptance* of the method, and finally the *process* of making the model.

- Preconceptions lie at the basis of all work. We humans are very often not aware of our preconceptions about things, and it is therefore difficult to describe them. The researcher finds it necessary to discuss some to these preconceptions, as they have caused heated discussion the model and its method. The first aspect to mention is that the modelling concepts are based on use of configuration software and are not meant for replacement of all programming. This causes misunderstanding, and participants in the test discussed lively some aspects where the difference in point of view was rooted in what each perceived to be included and what not. This ties well together with another discussion rooted in different worldviews / knowledge bases that often arose. Two such discussions are the use of AS-IS to get to TO-BE, and the inclusion of the "SuD" in the system. In the former, using AS-IS description to get to TO-BE states, an analysis of the current situation leads to improvement suggestions. This is generally against common praxis in Software Engineering (SE) and hence creates heated discussion on validity. The other is the inclusion of the system-under-development (SuD) in the system picture. This is not practiced in SE, but the product development domain uses this extensively (as crystallized in the AS-IS and TO-BE modes of working). Again, this was a cause

for lively discussion. To conclude, the different worldview or knowledge bases of the participants cause misunderstandings that require a lot of discussion to get through. It is hence very important to state at the beginning what is included and meant by certain terms and concepts.

- Acceptance of the general concepts suggested is apparent with participants. Of course there are issues that have to be solved, and misunderstandings that have to be corrected, but there is a general acceptance of the method and its possibilities. This makes the researchers hopeful.
- Again, the process is of interest. Here, several things showed up unexpectedly. It was always the researcher's belief that the suggested method was top-down in its approach. This turns out to be only partially true as the process is both top-down (as when making the SBM) and bottom-up (when constructing the EM). So, it could perhaps be called a "middle-in" approach in contrast to top-down. With the different knowledge bases, it is a time-consuming process to make the models, mainly because of the constant sidetracking experienced at the workshop. As mentioned earlier, it is very important to establish the right mind-set before starting the model work, as it can hopefully reduce sidetracking very much. A note by one participant to the researchers is worth mentioning here. It is important to differentiate between interfaces between models (this could be called a protocol for information exchange) and in handling information internally in EM (or the rules and their impact on cascading effects). This is of course connected with the model types. It is also worth noting that most effort has been put into developing the EM concept and only very little into the CM and SBM.

The last comment on this test could be a note on the next steps needed to validate method. As mentioned in earlier comments, an IT support tool is needed to fully make use of the method and its model. A complete test of such a tool would therefore be advisable. And of course, to finally prove this method's worth, a product system prototype with embedded configuration based on the method must be made. The prototype would help in testing the programming capabilities of the method and subsequently the exporting of a model from the current system.

This concludes our report of the four tests that were conducted. The highlights of the results are that predefined relations and decomposition techniques have to be suggested, and the process has to be supported with an adequate IT tool. Now, let us move on to the guidelines for model making.

# Chapter 7

# GUIDELINES FOR MODEL MAKING

The account of testing reported in the previous chapter points out quite clearly that some very plain guidelines for the model making are required. In this chapter, we try to produce such guidelines from the experience gathered so far. First, let us describe the example, a home entertainment system that produces both sound and pictures, a home cinema. Along the way, we use the example as walk-through case on equal footing with the case company models (created in the tests). This is because some are good for simple things, while other have complexity.

## 7.1 HOME ENTERTAINMENT SYSTEM

To clarify the modelling concept, let us look at a system that most of us can relate to, the reproduction of a moving picture and sound at home. The system needed to show a movie at home is a system of almost completely independent devices that have to work together to generate the sounds and pictures of a movie. Let us go through the exercise of making the three models and see how it goes. We construct the three models mentioned earlier, based on this example.

### 7.1.1 THE SBM – GENERAL SYSTEM DESCRIPTION

First step is to decide what the system is. What archetypes are needed to implement the required functionality to fulfil the overall purpose of the system? For pedagogical reasons, this will be very detailed with a lot of rationale explained along the way.

A first look at a home entertainment system (HES) shows a combination of industry standard modules with standard interfaces (RCA, coax etc.) that are guaranteed to fit between all. The evolution and history of stereo modules is covered in depth in (Langlois & Robertson 1992). More recently, HES has been expanded to cover two main applications: to produce vision (or moving pictures) and sound. These two applications are actually completely independent of each other, even though they are very often realized in integrated products. Let us use this modular system to demonstrate the models in Kefec.

To simplify the process and allow for greater understanding of the problematic involved, we use common lingua to describe the environment and the elements therein. This means that we do not enforce the standardization part of the CM but use a domain lingua, a strange mixture of function terms, trademarks and standards. Finally, we redo the models with standard lingua as suggested in CM.

A good starting point is the environment evaluation shown in Table 40

**Table 40 – Task environment for home entertainment**

| Agent type | **P**erformance measure | **E**nvironment | **A**ctuators | **S**ensors |
|---|---|---|---|---|
| Home Entertainment System | Sound quality, Vision quality | DVD, HCR, Speakers, TV, cables, disks, users | Selections | user preference, hardware detection |

The two applications result in two performance measures, but they are not described in detail; it is only stated what is good, better and best. The system relies on only two sensors and can only use one actuator to select between different possibilities.

A general description of the system is as follows: There are three archetypes in the system, the DVD player, the Home Cinema Receiver (with speakers attached), and the Television. These are connected through different connection types as shown in Figure 125.


**Figure 125 – Home entertainment system setup**

These three devices, connected together in a proper way and configured accordingly, deliver the final product or experience to the customer, the pleasure of watching a movie at home. Each of these devices has parameters that deal with both internal and external environments that have to be set. To better understand this system, let us make a general functional abstraction of the system. The system has two applications that are realized with two separate function chains as shown in Figure 126.

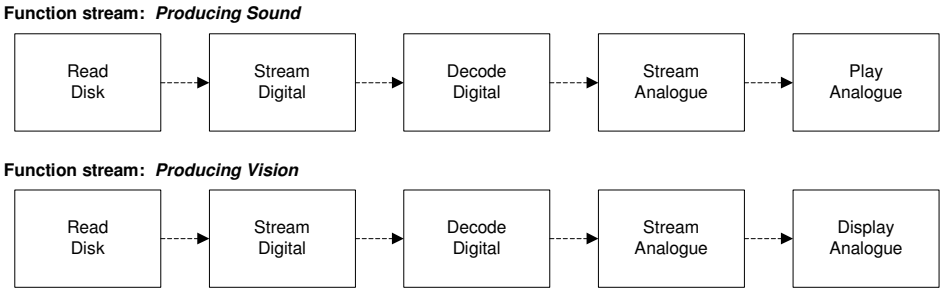



**Figure 126 - Main functions as function streams**

Here, we have "hidden" the amplification part, which could also be considered a main function, but for demonstrative purpose it is not necessary as in this case, amplification is included in *Stream Analogue*. The three archetypes supply these functions; the last function of producing sound (*Play Analogue*) is also omitted, as the speakers add nothing to the example.

This functionality is realized as follows: DVD player can read, stream and decode; the TV can display and play; the HCR can decode and stream. A graphic presentation of which does what is shown in Figure 127.

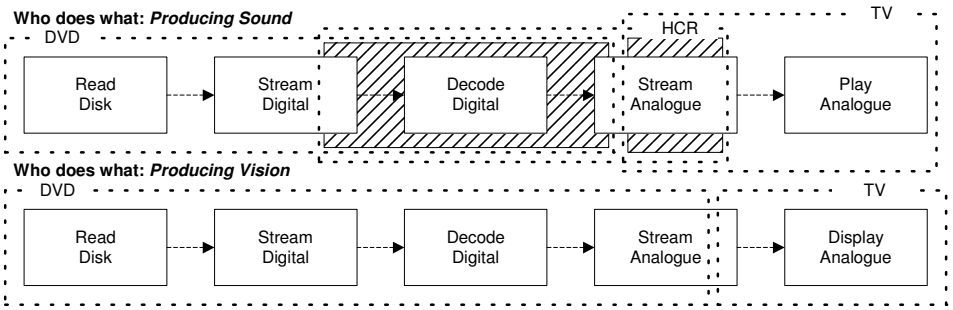



**Figure 127 - Which does what in HES**

Observe that both DVD and HCR can decode, and both TV and HCR can amplify; these dualities are shown highlighted in Figure 127. The functions are realized with different standards. Decoding is done in different qualities (bit rates) and on a different number of channels. There are decoding in two channels for stereo and SACD, where stereo has a lower bit rate (and hence quality) than SACD; and then there is six-channel decoding for Dolby Digital 5.1, DTS and SACD, where Dolby is the lowest and SACD the highest quality. A summary of which device does what is given in Figure 128 and Figure 129.

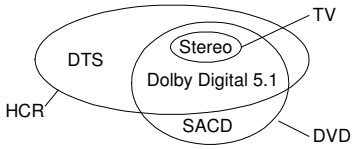



**Figure 128 - Sound reproduction**

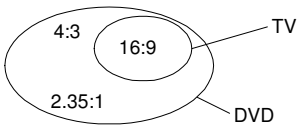



**Figure 129 - Vision reproduction**

In this setup, the HCR has nothing to do with vision reproduction, while all devices are included in sound. So, let us consider a setup where the receiver has internal DTS and Dolby Digital 5.1 decoding but no SACD decoding, while the DVD player is equipped with internal decoding of Dolby Digital 5.1, SACD and Stereo. Then, the two devices together can span many of the possible sound systems for both movies and music. The TV can also play stereo sound, but this is actually redundant in this setup, as shown by the overlap in Figure 128. The system breakdown is thus as shown in Figure 130.

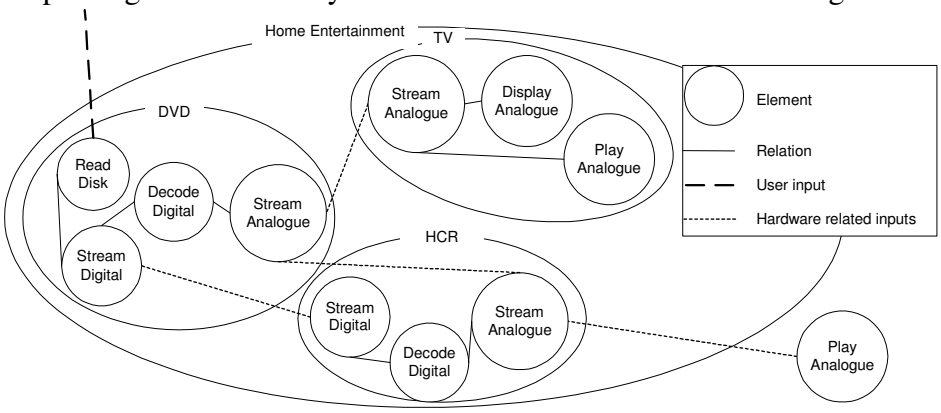



**Figure 130 - SBM for Home Entertainment**

Now, we have the general system breakdown and its main functional elements. Notice that the *Play Analogue* function is outside the system boundary, as this is the speaker and has no variance. For each subsystem in the SBM, we make an encapsulation model where details regarding the configuration are shown. This is presented in the next section.

### 7.1.2 ENCAPSULATION MODELS FOR THE DEVICES

Again, to clarify, the configuration sets a set of parameters to certain values. Most of these have to do with hardware selection or which path of hardware is to be used. For this example, Table 41 shows a list of parameters that are to be configured in the devices.

**Table 41 - Parameters in each device**

| Device | Parameters | Int | Values |
|---|---|---|---|
| DVD player | Movie format | | 4:3, 16:9, 2.35:1 |
| | Sound systems compatible | | Stereo, Dolby Digital 5.1, DTS, SACD |
| | Sound decoders | | Dolby Digital 5.1, SACD (2 ch and 5.1 ch) |
| | Preferred language | Int | 1 selected or original |
| | Preferred subtext language | Int | 1 selected or none |
| | Subtext placement on screen | | Bottom of screen, 5% raised, 10% raised, 15% raised |
| | Output type to TV | | Scart, S-video, Component |
| | Output type to HCR | | Digital, Analogue 5.1 ch, Analogue 2 ch |
| | Device on | | on, off |
| Home Cinema Receiver | Sound decoders | | Stereo, Dolby Digital 5.1, DTS |
| | Device on | | on, off |
| | Input mode | | CD, DVD, Cassette, Record |
| | Input type | | Digital, analogue 2 channel, analogue 6 channel |
| Television | Device on | | on, off |
| | Volume | | mute, 0 … 100% |
| | Movie format | | 16:9 |
| | Smart stretch | Int | 4:3 => 16:9, 2.35:1 => 16:9 |
| | Sound decoders | Int | Stereo |
| | Input type | Int | Scart, S-video, Component |
| | Screen size | Int | 32” |
| DVD disk 1 | Movie format | | 2.35:1, 16:9 |
| | Spoken language | | English Stereo, English Dolby Digital 5.1, English DTS |
| | Subtext language | | English, Danish, Swedish |
| DVD disk 2 | Movie format | | 4:3 |
| | Spoken language | | English Stereo, English Dolby Digital 5.1, Icelandic Stereo, Danish Dolby Digital 5.1 |
| | Subtext language | | English, Danish, Swedish, Icelandic |
| SACD disk 3 | Music | | SACD 2 channel, SACD 5.1 channel |

Note that only relevant items are listed in Table 41 for the clarity of the example; in reality a lot more parameters could be set in each device.

Some notion of quality functions has to exist to evaluate the best solution. In this case, the measure of quality is actually quantitative: the sound quality is given by bit-rate of decoding, where higher is better; and vision quality is given by width of screen, and here higher is also better. The quality is given in the EM as priority of choices, so if more than one is available, the higher (highest) quality is chosen. For example, DTS sound is better than Dolby Digital 5.1, which then again is better than stereo. Let us now draw an EM for the three archetypes, here with an example of each where the values are listed in Table 41. The first device is the HCR, as shown in Figure 131.

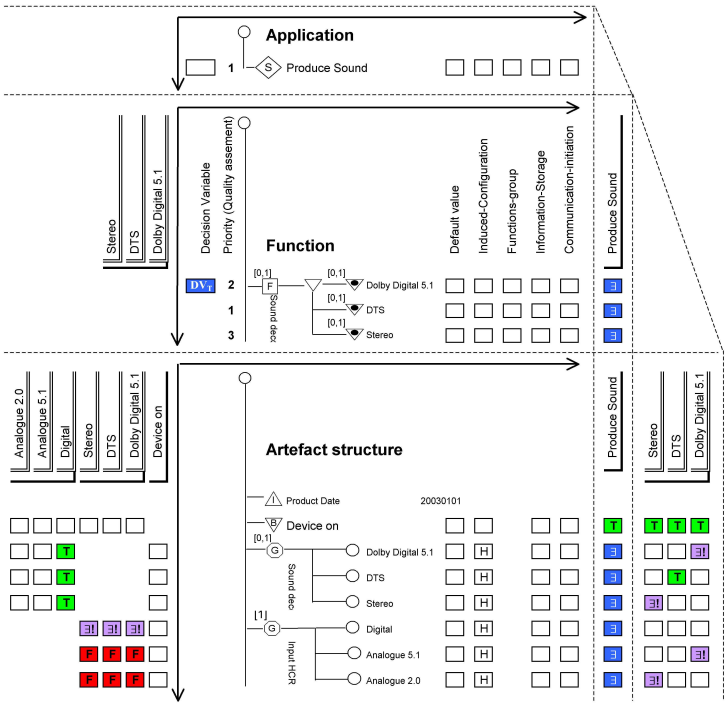



**Figure 131 - Encapsulation model for HCR**

As seen in Figure 131, there is only one application in HCR, the production of sound. This is realized with a single function with three choices. In the HCR, there is only one tentative DV. The next EM for a device is the TV, as shown in Figure 132. The TV has two tentative DVs, one for each application.

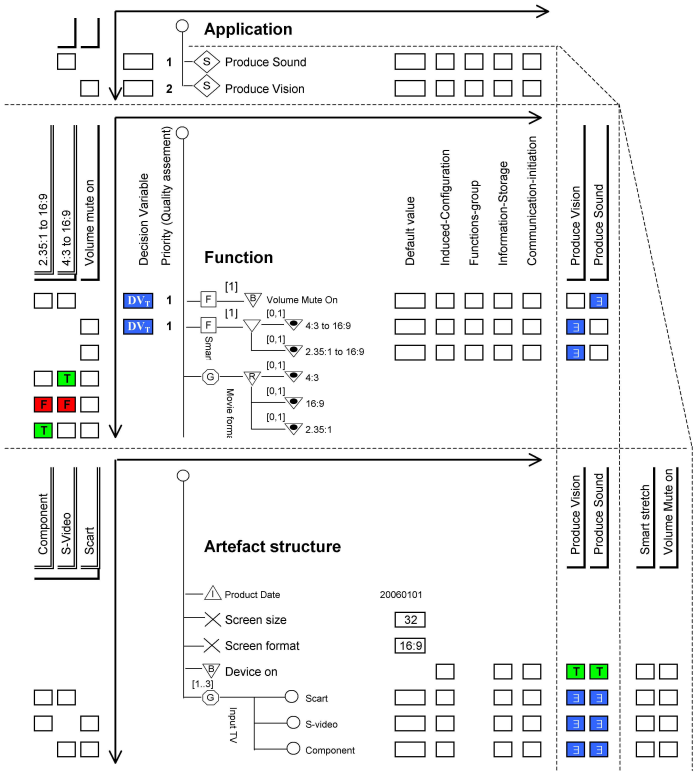



**Figure 132 - Encapsulation model for TV**

The next EM is the one for the DVD. This is the biggest one, as the DVD is the centrepiece in the system and is actually the "controller". The DVD has both

applications, sound and pictures, and it can run both, or one at a time. The EM for DVD is shown in Figure 133. The DVD has four tentative DVs, where one is for sound and three are for pictures.

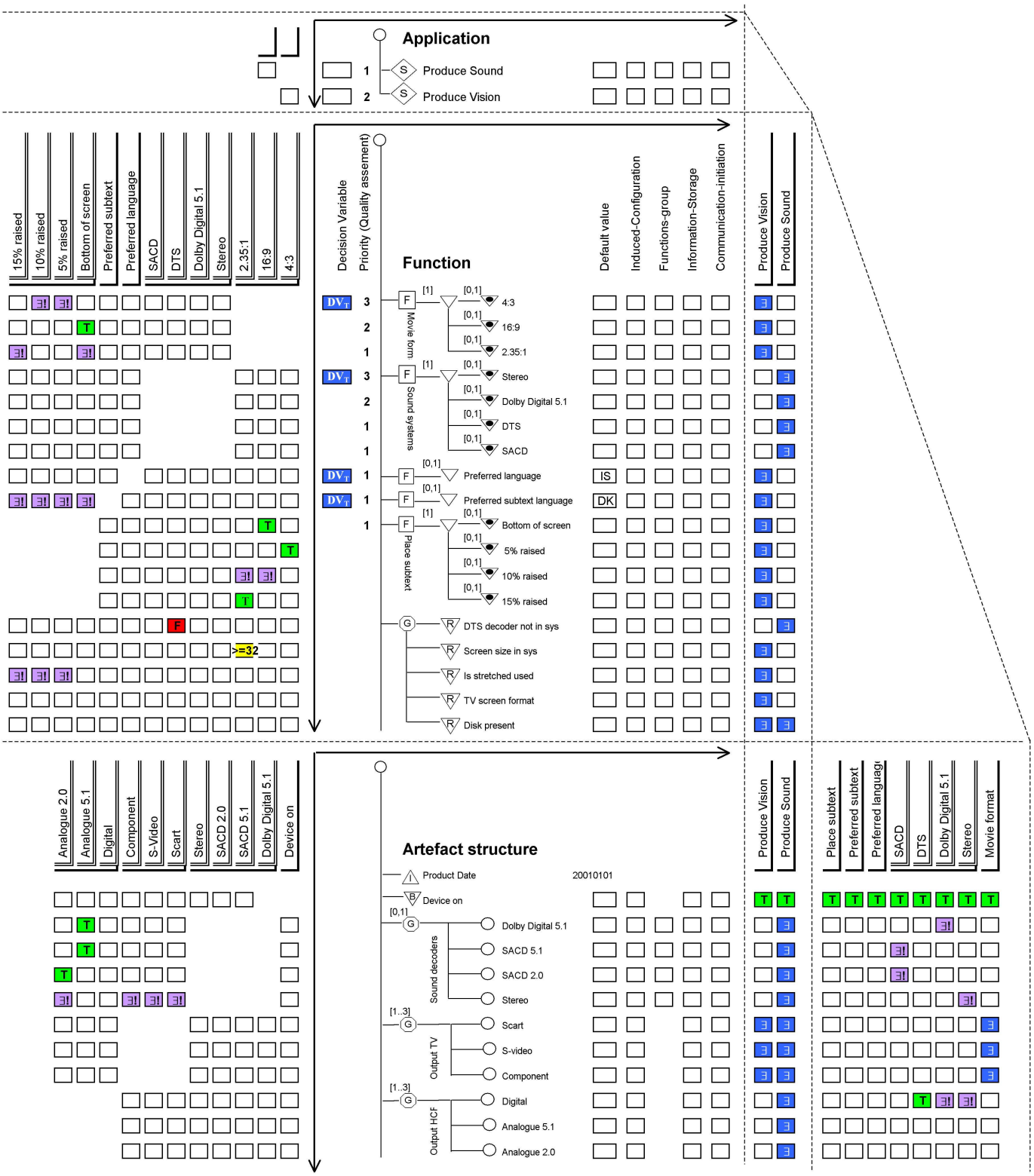



**Figure 133 - Encapsulation model for DVD**

The three devices actually rely on the fourth one, the disk, to complete the hardware setup. The disk also involves user input. The only thing that the user has to do is insert the disk into the player. In doing so, the user also has "closed" the solution space and

given the system all the information it needs to complete the setup. We therefore need to model an EM for each disk so we can get a description of all subsystems in order to make the stacked EM for a single scenario later. EM for the three disks is shown in Figure 134 to Figure 136

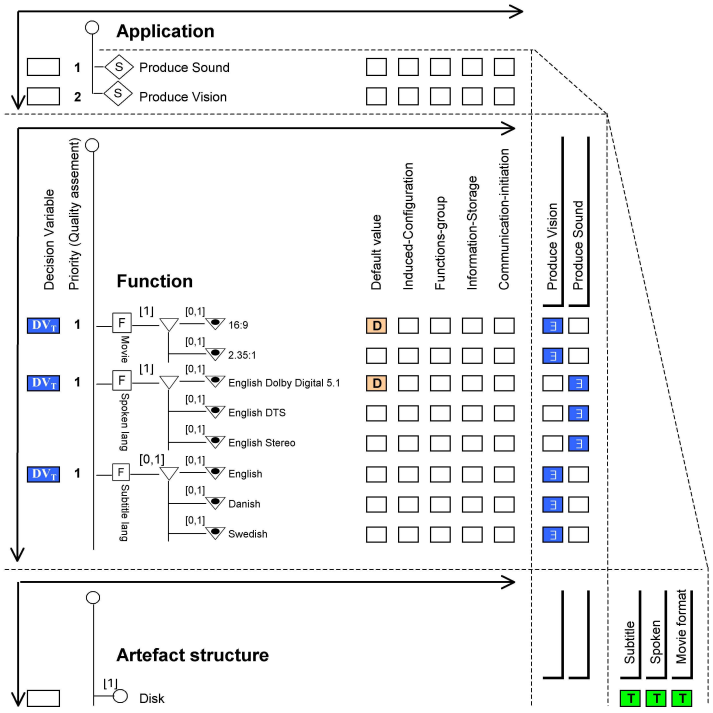



**Figure 134 - Encapsulation model for Disk 1**

These six EM models together form a solution space, from which the user can enjoy an application. There are no DVs but the application is deduced from the inserted disk. Here, items like setting sound volume are left out and considered operation. Let us now look at the communication model to define how the devices should communicate.

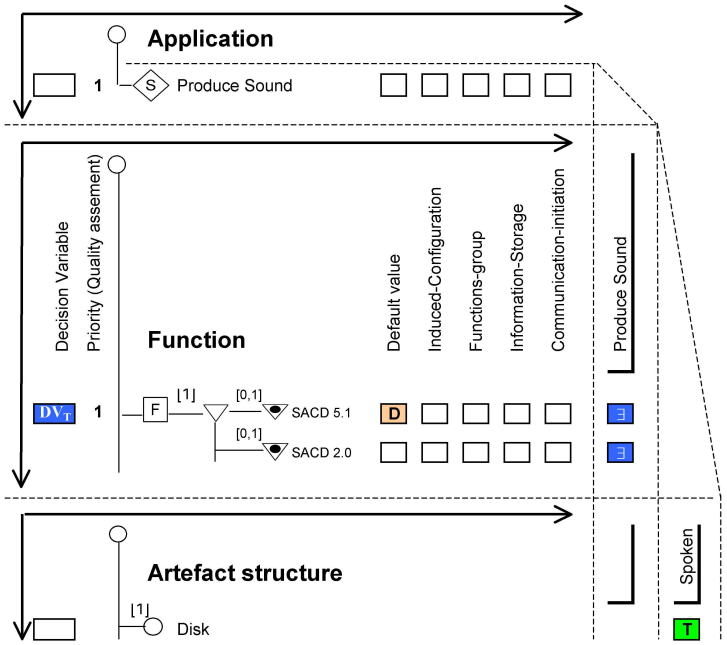



**Figure 135 - Encapsulation model for Disk 2**

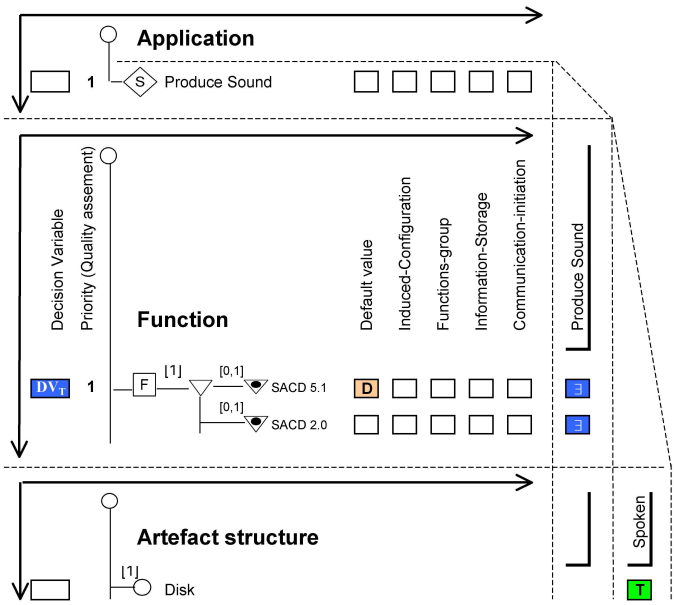



**Figure 136 - Encapsulation model for Disk 3**

### 7.1.3 THE COMMUNICATION MODEL

The CM defines how the devices exchange information and by which means. Here, the criterion is that the system needs to produce sound and vision with maximum quality. These quality evaluations are stated as a set of rules in this example. As mentioned previously, information needed to decide on quality is coded as priorities in the EMs. The communication and its consequent negotiation has to solve routing, which does what, possible solutions (solution space), preferences and other relevant issues to maximize the goal of quality function. Table 42 shows the rules that dominate this example. The type column (T) in Table 42 refers to hardware-induced (H) or application-induced (A) configuration.

**Table 42 - Rules for the system**

| # | T | Rule logic (description) | Concerning |
|---|---|---|---|
| 1 | H | DVD player decoder can only be used if HCR is connected with analogue 5.1 channel and/or 2 channel | DVD, HCR |
| 2 | H | HCR decoder can only be used if DVD is connected with digital channel | DVD, HCR |
| 3 | H | DVD player has to be connected to HCR with at least one route | DVD, HCR |
| 4 | H | DVD player has to be connected to TV with at least one route | DVD, TV |
| 5 | A | DTS is better than Dolby Digital 5.1 | DVD |
| 6 | A | Dolby Digital 5.1 is better than Stereo | DVD |
| 7 | A | 16:9 is better than 4:3 | DVD |
| 8 | A | For all ratios other than own (16:9), use smart-stretch to fit to screen | TV |
| 9 | A | For screen size larger (or equal) than 32”, use highest ratio | DVD, TV |
| 10 | A | SACD is better than Stereo | DVD |
| 11 | A | Use fewest devices possible | DVD, HCR, TV |
| 12 | A | Newer is better | DVD, HCR, TV |

Error handling has to exist to check if devices are correctly connected via different input types and to tell if there are no possible solutions. The system should not allow

illegal solutions. A way to visualize the connection between devices is the parameter diagram (*par* diagram) in Figure 137.

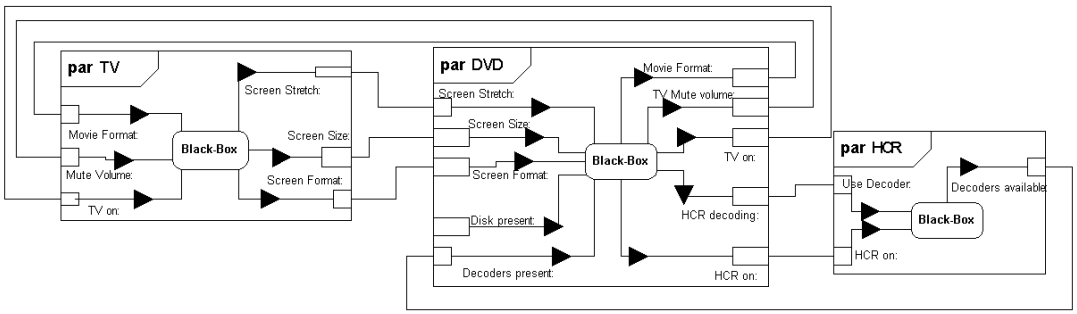



**Figure 137 - SysML parametric diagram for home entertainment**

The diagram only shows parameter exchanges needed for the application phase. The hardware connection parameters are not shown. This is done to simplify the view. A separate *par* diagram could be drawn for that. The actual hardware connections are shown in Figure 125.

The main purpose of the CM is to facilitate automatic generation of the stacked EM. Since we make the stacked EM manually, the CM does not need to be so specific. When we revisit this example with a standardized lingua, we remake the CM with a lot more details. Let us look at the stacked EM.

### 7.1.4 STACKING EM FOR THIS EXAMPLE

For the hardware present in this example, the stacked EM shows the possible solutions. Let us draw one stacked EM for the combination of DVD, HCR and TV, and then three separate EMs with the three disks included. The latter three are shown in the discussion of the scenarios in the next section, but let us first look at a merger of the three EMs for DVD, HCR and TV as shown in Figure 138.

This stacked EM shows ALL the possibilities that the system can handle in regard to the different disks. Inserting a disk works as a "filter" on this solution space and narrows it even further, and with the rules and priorities in place, the system should be able to select "the best" solution available. The following are some scenarios that describe how the system should react to different inputs – here, DVD disks with different information encoded. Now, let us look at the scenarios.

### 7.1.5 SCENARIO DVD DISK 1

The DVD player reads disk 1 and defines the possible setups for sound and vision. Let us start with vision. As the disk has both 16:9 and 2.35:1 format, it has to ask the system which to use. The DVD player turns on the TV and asks how big it is. The TV answers 32 inches, and 2.35:1 format is chosen. Next step is to decide if subtexts are to be used. If they are, the DVD player has to ask the TV whether some picture processing is being done (smart stretch etc.) so that the subtext has to be placed on a different part of the screen. The TV answers to put the subtext onto the picture so that it is not cut out when the picture is stretched to fit the 16:9 screen, which means that the DVD has to raise the subtext by 15%.

Then comes the sound part. As the disk has only one language this is chosen by default, but there are three different standards with different sound quality.

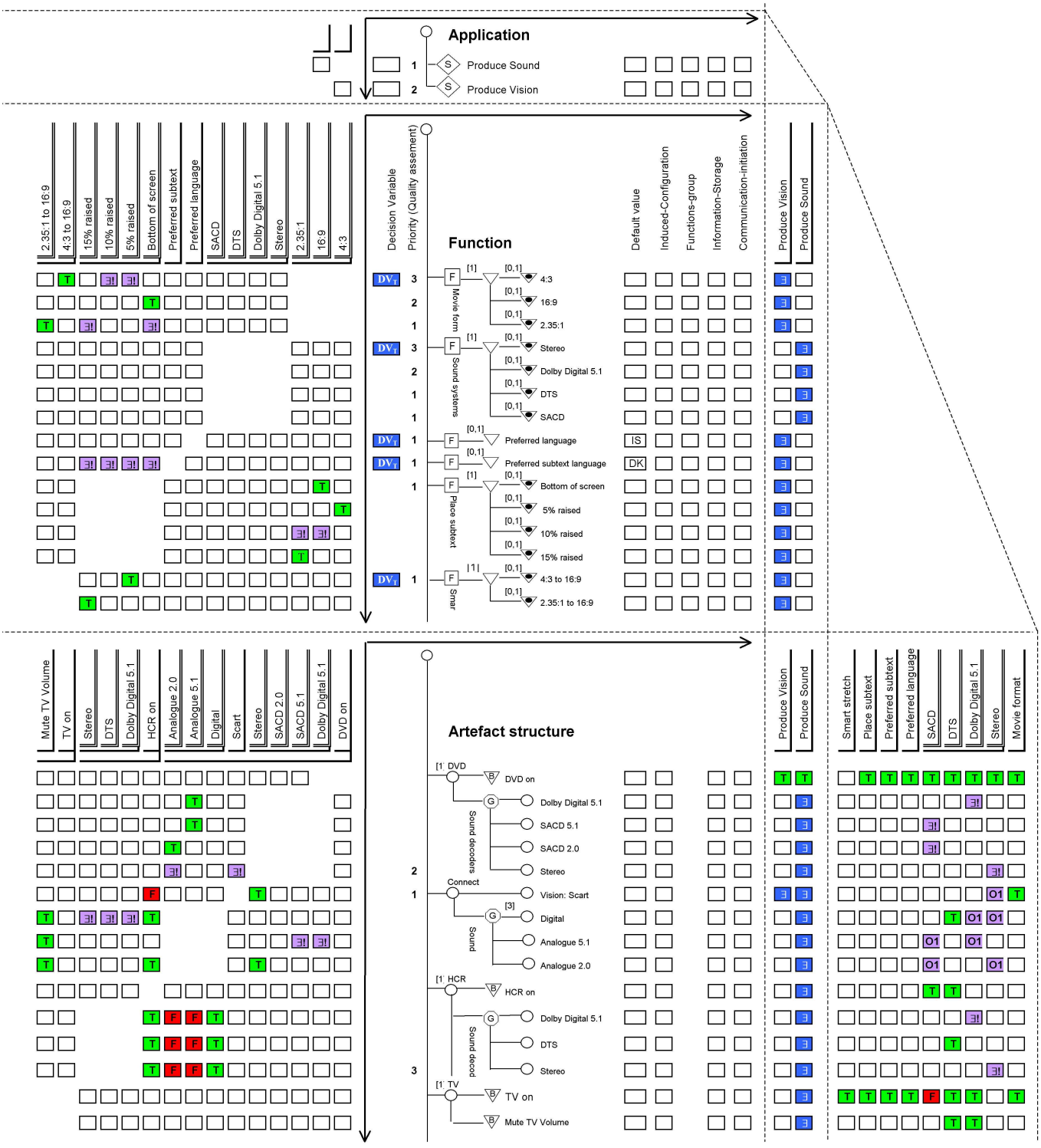



**Figure 138 - Stacked EM for home entertainment**

DVD player knows that DTS is best but cannot handle the decoding itself. It therefore asks the other devices (note: both the TV and the HCR) if they can decode DTS. The HCR answers that it can. To be able to use the decoder in HCR, the signal has to be sent via the digital channel. The player turns on the HCR, sets it to decode DTS from digital channel, and then passes the DTS signal on. As the sounds are being reproduced

with the HCR, the DVD player tells the TV to turn down the volume or set to mute. The route of the decision making described is shown in the Figure 139, where the number one is first selected and so on. Grey areas are the values not possible when the disk is combined with the existing hardware.

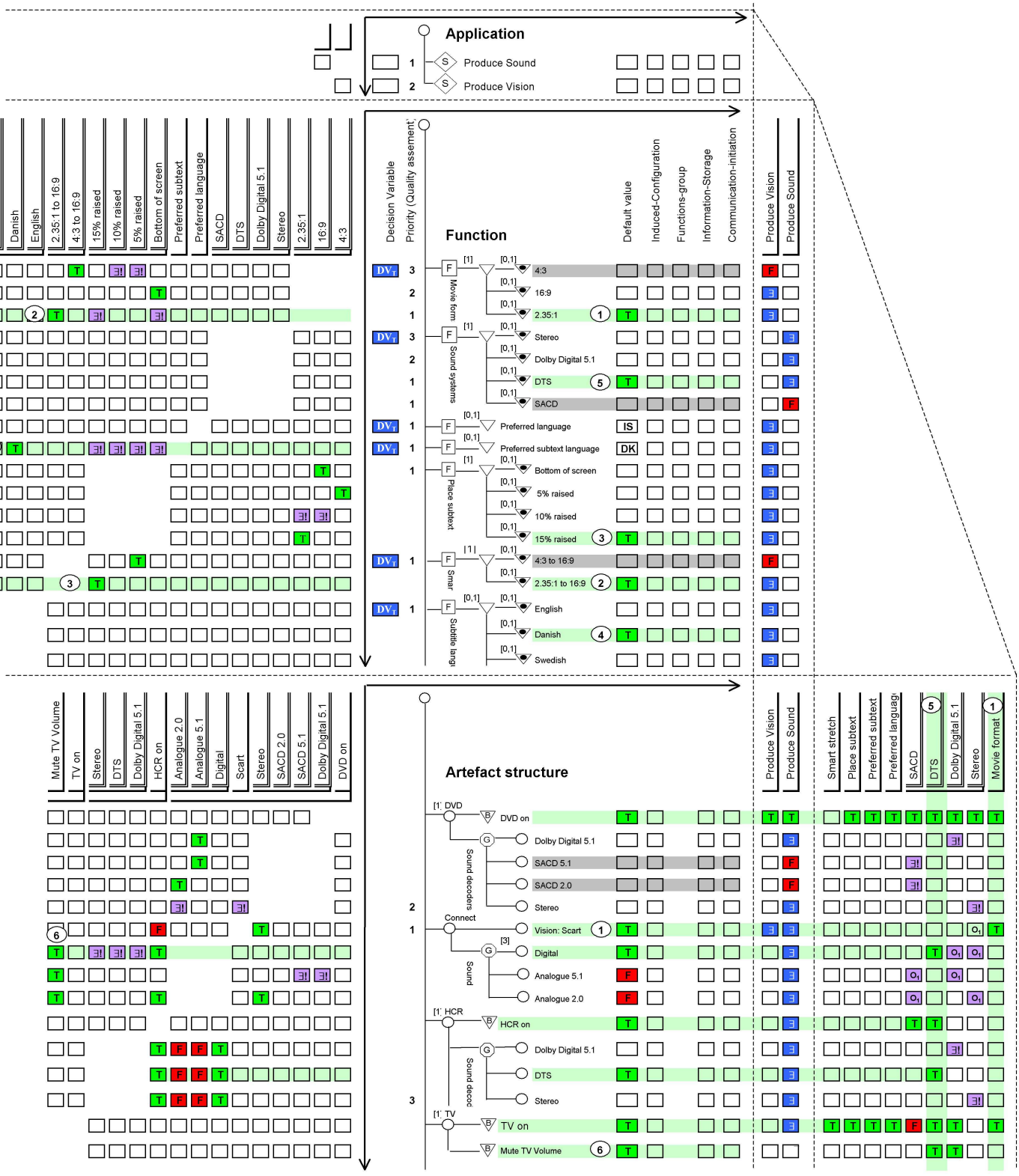



**Figure 139 - Scenario 1 - Stacked EM and decision route**

### 7.1.6 SCENARIO WITH DVD DISK 2

After the player has read the disk, it can decide on a plan of action. Again, let us start with pictures. Since the disk only has 4:3 format, that is used. Next step is to decide if subtexts are to be used. If they are, the DVD player has to ask the TV if some picture processing is being done (smart stretch etc.) so that the subtext has to be placed in a different part of the screen. The TV answers to put the subtext 5% higher into the picture so that it is not cut out when the picture is stretched to fit 16:9 screens.
Here, the preferred language controls the sound. With Icelandic as preferred language, this would mean that only stereo is available. Using the rule that minimum amount of devices should be used (rule #11 in Table 42), the player does not turn on the HCR but uses the TV for sound reproduction.

### 7.1.7 SCENARIO WITH SACD DISK 3

The player reads the SACD disk and decides that no visual processing is needed. Therefore, the TV is not turned on.

As the disk contains both stereo and multi-channel sound, the player chooses the higher number of channels, as it tries to interpret rule #6 in Table 42 for music. The DVD player has internal SACD decoder and it knows that the signal is then sent via analogue channels to HCR, so it turns on the HCR, checks if 5.1 analogue channels is available, and then sets HCR to receive via 5.1 analogue.

This concludes the example. Now, let us go through the making of such models step by step and try to highlight areas that need attention, and give some guidelines on the structuring process.

## 7.2 SYSTEM UNDERSTANDING

The first step is to gain an understanding of the overall system in question. This probably means that we have to take a "step back" and try to find the main purpose (or purposes) of the system. Think about the above-mentioned example: what is the purpose of the system? Try to think in general terms and close to the customer. When you are proficient in some domain, it is easy to get caught in the details and start to look at technical solutions instead of purpose. Stay on the high level! Researchers find it very helpful to "visualize" a journey where you place yourself as part of the "flow" through the system. What happens to you on the way? Where are additions and where does one have to supply something? Thinking about the home entertainment example, place yourself with the "data" on the disk and "flow" through the system to the final "station", either the screen or the loudspeaker. This exercise of course requires a very good understanding of what is going on, but let us assume that this is no problem. Such an exercise should produce the functional streams.

### 7.2.1 FUNCTIONAL STREAMS

Functional streams are rooted in the generally accepted notion that functions can be described as verb-noun combinations to describe transformations of flows. What differs here is the abstraction level used. The functional streams are not detailed axiomatic-like descriptions, since that would replicate *Functional Basis* and other similar methods. The key here is *effect*¸ as understood by (Hubka & Eder 1987). Focus is on what the product is truly intended for and how that relates to the customer's needs. We should be able to describe a product on such a high level in relatively few streams, each only building on several elements. Think about the home entertainment example: two streams, each with five elements (see Figure 125 on page 155). The streams have the "same" element names, as the process is similar. But when implementing, decoding sound is not the same as decoding pictures. So, guidelines for making the functional streams in the form of questions could be:

- What happens to the stream from start to end? In the example: the sound stream starts as a zero-one chain in the form of no-holes / holes on the disk, equalling digital 0-1s.
- What are the main transformations that the stream receives? In the example: Changing holes / no-holes to 0-1s, changing 0-1 to waves (analogue), amplifying waves, changing waves from electronic to pressure, and moving the stream from one change to the next.
- What are supporting attributes and should these probably not be included in the streams?
- Which attributes and transformations should be made into elements in the functional stream? Which should be hidden within others? In the example: amplifying is embedded within stream analogue.

Once the streams are completed, they have to be mapped to the archetypes to form the system breakdown model (SBM). This is the summary model of all system understanding and is to be used further down the line.

### 7.2.2 SYSTEM BREAKDOWN

As the SBM is an overlapping FS on archetypes, it should not be hard to make. The things to remember here are overlapping FSs. Think again about the example: the first element in both streams is *Read Disc*. This is actually the same element, whereas the second element *Stream Digital* is actually two different actions, as sound and vision are not treated equally in the system. Guidelines for making SBSM:

- Are there overlapping elements in the FSs?
- What relations exist outside of those drawn in the FSs?
- Are placements of elements from FS possible in many archetypes?
- What lies within the system? Are all elements in the FS inside the boundary?

Remember that the purpose of the SBM is to create a platform to work on the EMs. It should help in determining how many different EMs are needed and suggest the first draft for decomposition or groupings. Let us now move on to the decompositions.

## 7.3 DECOMPOSITION GUIDELINES

The first step is to make the Encapsulation Model, and the first step of the EM is the PVM centrepiece. The PVM is dominated by two actions, splitting of elements to three abstraction levels and the decomposition within each level. This section tackles these two actions, first on a general level and then in more detail. First thing on the list is the separation of elements into the three abstractions.

### 7.3.1 ABSTRACTION LEVELS PLACMENT

This section deals with how it is possible to place elements in the different abstractions. At the end of the section, we present a list of guiding questions to aid in the process.

The first step is to list all the variables and their values. The data set used in this project was based on variables and values extracted from the configuration software at the case company and a printed list of rules. Other forms of data sets can also be used. The form and structuring of data does not need be in a specific format as long as the variables, values and most relations between them can be extracted. Any remaining relations can subsequently be added in the process, but it is not necessary in the initial step. This step should be seen as rough sorting of elements.

| Variables | | | | | | Program screen no. | Value | PVM (A/F/S) |
|---|---|---|---|---|---|---|---|---|
| Variable level no. 1 | Variable level no. 2 | 3 | 4 | 5 | 6 | | | |
| Number of pumps | | | | | | 1.1 | [1-6] [integer] | S |
| Level sensor type | | | | | | 1.1 | Analog level sensor/ Float switches | S |
| | Number of float switches | | | | | 1.1 | [0-5] [integer] | S |
| | Other switch function | | | | | 1.1 | no function / Dry running / High Level / Dry runnning & High level | F |
| | | | | | | | | |
| Analog input A1 | | | | | | 1.2 | on | F |
| | Input function | | | | | 1.2 | Level | F |
| | Sensor range | | | | | 1.2 | [0-635.65] [m] | S |
| | Sensor zero offset | | | | | 1.2 | [0-635.65] [m] | S |
| | Signal type | | | | | 1.2 | 0-20mA / 4-20mA / 0-10V / 2-10V | S |
| Analog input A2 | | | | | | 1.2 | on / off | F |
| | Input function | | | | | 1.2 | Flow, discharge / power / Level overflow / Flow overflow | F |
| | Sensor range | | | | | 1.2 | [0-6553.5][l/s] | S |
| | Sensor zero offset | | | | | 1.2 | [0-6553.5][l/s] | S |
| | Signal type | | | | | 1.2 | 0-20mA / 4-20mA / 0-10V / 2-10V | S |
| Pulse counter | | | | | | 1.2 | on/off | S |
| | Pulse counter function | | | | | 1.2 | Volume, discharge / Energy / Flow overflow | F |
| | Pulse counter scaling | | | | | 1.2 | [0-655.35][count/unit] | F |
| Digital input logic 1 | | | | | | 1.2 | NC / NO | F |
| Digital input logic 2 | | | | | | 1.2 | NC / NO | F |
| Digital input logic 3 | | | | | | 1.2 | NC / NO | F |
| Digital input logic 4 | | | | | | 1.2 | NC / NO | F |
| Digital input logic 5 | | | | | | 1.2 | NC / NO | F |
| Digital input logic 6 | | | | | | 1.2 | NC / NO | F |
| Foam draining | | | | | | 2.1 | on / off | A |
| | Foam drain level | | | | | 2.1 | [??-??][m] | F2 |
| | Foam drain, stop delay | | | | | 2.1 | [??-??][sek] | F2 |
| | Foam drain, start interval | | | | | 2.1 | [??-??][min] | F2 |
| DO7 function | | | | | | 1.2 | Not used / All alarms & warnings / All alarms / Urgent alarms / High water alarm / Mixer control | F |
| | Ack. mode | | | | | 1.2 | Auto / Manual | F |
| DO8 function | | | | | | 1.2 | Not used / All alarms & warnings / All alarms / Urgent alarms / High water alarm | F |
| | Ack. mode | | | | | 1.2 | Auto / Manual | F |
| | | | | | | | | |
| Use Analog Input | | | | | | 1.3 | on / off | F |
| | Input function | | | | | 1.3 | Water in oil / Input current | F |
| | Sensor range | | | | | 1.3 | [??-??][%, A] | F |
| | Sensor zero offset | | | | | 1.3 | [??-??][%, A] | F |
| | Signal type | | | | | | 0-20mA / 4-20mA / 0-10V / 2-10V | F |
| | | | | | | | | |
| Mains frequency | | | | | | 1.4 | [50, 60][Hz] | F |
| PU 102 installed | | | | | | 1.4 | on / off | S |

**Figure 140 - Placing variables in abstractions**

All of the variables are reviewed in order to determine which level of the PVM they belong to, i.e. *Application (A)*, *Function (F)* or *Artefact (S),* and the level is listed for each variable. The review may show that variables that are already placed in some structure in the original data source belong to different levels in terms of the PVM

structure. Using different colours for each PVM level, as seen in Figure 140, can emphasize this.

The example shown in Figure 140 is from the list generated for the case company. The variables are listed to the left. The variables are structured according to the way it is done in the original data source, the configuration program *Grundfos PC Tool Modular Controls*. The variables in the tool are grouped to different screens wherein small groups are generated that show hierarchical structure. If the variables are part of a hierarchical structure, they are placed in different columns according to their placement in the hierarchy (the columns *variable level no.1*, *variable level no. 2* and so on, shown in Figure 140). Answering the following questions can make the assignment of a variable to a PVM level:

- Application level assignment
  - The variable describes a user need, such as a functional or physical need, but not a selection of physical components.
    - Functional need: For example, the need for an overall alarm level that can be set to a value of high, medium or low
    - Physical need: For example, the need for a pit depth of a certain size to which all other depth or height dependent variables are related.
  - The variable works as a communication between the user and the system across the system boundary, i.e. the variable does not directly induce a change in the system, but only through its effect on other variables.
- Function level assignment:
  - if the variable describes functionality of physical components or behaviour of the system that is not hardware related. These can be on two levels, technical and effect-like.
- Artefact level assignment:
  - if the variable describes a choice of physical components.
  - if the variable describes a hardware setup.

Once variables have been assigned to a specific level, there are three "groups" of variables that have to be structured. It is very possible and actually quite probable that the application level is empty in analysing an existing product. This should just be seen as an opportunity for a redesign. Let us look at this structuring, which is known here as decomposition.

### 7.3.2 DECOMPOSITION

Decomposing or making parts out of a whole is something humans are quite good at. We do this all the time to deal with complexity. The problem is that we do not notice it, nor can we state how we go about it. The same holds true for the work of analysing an artefact structure with regard to its software. Depending on the source of the variables, there may already exist a structure with a specific order or grouping. The groupings of variables can be a result of pure programming concerns, the way they are used or in concordance with the physical component from which they originate. These

ways of grouping bear similarities with the three abstraction levels and are appropriate at the right level.

The decomposition literature (see Figure 79 on page 90) suggests (as one of many) that there are three main ways of decomposing: external (criteria), internal (portion) and no structure (pieces). The "no structure" is of no interest here and is subsequently left out. The other two, on the other hand, are very useful. Let us explain. Decomposition driven by internal properties is reliant on the actual artefact construction, so a modular decomposition is an internal one. An arbitrary decomposition that is not reliant on the actual artefact, like customer needs decomposition, is an external criterion. We could thus state: use internal decomposition to make the artefact level, an external decompostion for the application level, and both for the functional level, as its purpose is to connect the other two. This duality in the functional level is apparent throughout this thesis. It is probably the reason why Functional Basis does not work as sole language for the functional level, since it is only driven by internal decomposition. And therefore it is hard to map to actual customer needs or the application.

To recapitulate, the first approach to decompose each level is:

- Use external criteria to decompose the application level.
- Use both ways to decompose the functional level.
- Use the internal properties to decompose the artefact level.

The following guideline questions can aid in the decomposition process:

- What are the external criteria that dominate the artefact and its environment?
- What are the internal properties that rule the artefact? Is it modular? Is it DFA (design for assembly)?
- How do the external criteria tie up to the sales view for the artefact?

Dennis Buede suggests some thoughts on common pitfalls in functional decompositions. One should avoid:

1. *Including the external system and their functions.*
2. *Choosing the wrong name for function. The function name should start with an action verb and include an object of that action. The verb should not contain an objective or performance goal such as minimize, but should describe an action or activity that is to be performed.*
3. *Creating a decomposition of a function that is not a partition of that function. A function should not be child of itself. The sub-function of a function should all be at the same level of abstraction.*
4. *Include a verb phrase as part of the inputs, controls or outputs of a function. Verb phrases are reserved for functions.*
5. *Violating the law of conservation of input, controls and outputs.*
6. *Trivializing the richness of interaction between the functions that decompose their parents.*
7. *Creating outputs from thin air. The most common mistake is to define a function that monitors the system's status but does not receive inputs about the functioning or lack of functioning of other parts of the system.* *p.204 in (Buede 2000)*

A special type of decomposition is the grouping activity. Let us look at that.

### 7.3.3 GROUPING VARIABLES

After the initial decomposition is done, a further breakdown or structuring can be achieved by grouping variables in the tree. There are several ways to do this. The groupings are of two main types, those that can be done directly in the tree and those that rely on having all the relations described. The following list summarizes these:

- Grouping that can be performed when constructing the tree (not reliant on having relations in place)
    - Reducing the level of detail to ease communication through a better overview
    - Identifying variable groups to form the basis of new decision variables
- Grouping that relies on relations matrixes:
    - Restructuring the PVM according to relations instead of physical or programming considerations
    - Reducing the level of detail to ease identification of relations patterns (this is particularly true when values are also excluded)
    - Increasing the understanding of the functionality of the product by seeing how seemingly unconnected parts are in reality connected through behaviour

A discussion of relationship groupings follows in the section on "Trimming Guidelines" on page 174.

### 7.3.4 FINDING DECISION VARIABLES

The main objective of this thesis' premise is to reduce decision variables. It is hence very important to find them. Remember that we earlier defined decision variable as an input that crosses the system boundary and on which the user has to take action. Remember also that even though a variable has a default value or is populated with file download, it is still a decision variable.

The process of finding decision variables is two-fold. We could say that it is the AS-IS and TO-BE situations, meaning that we first find the current DVs and then suggest what are relevant DVs to use in the future. Some decision variables are internal, meaning information that can be shared within the system boundary, if subsystems could communicate. This ties to the hardware- and application-induced configuration-sorting mentioned in the next section.

Guideline questions to aid in finding decision variables:

- Is the variable internally set, within a sub-system, within the system?
- Where does the information needed to set the variable come from?
- When is the variable set – in communication phase, negotiation phase or even operation phase?
- Does the variable ever change value? Should this be a variable?

It is probable that finding DVs is hard in the opening stages of making the models. Finding AS-IS is probably relatively easy but suggesting the TO-BE is not. The process of finding DVs should be expected to be very iterative, and all DVs will probably first be found when the models are completed. A good aid in finding the wished state is to sort the variables.

### 7.3.5 VARIABLE SORTING

We like to introduce several columns of information to the right of the PVM tree, as a helpful step for decision variables, functional structure and the communication model. These columns serve as aids and can be removed after they have served their purpose. Let us look at the four columns that deal with the three groups mentioned above.

Two of the columns deal directly with DVs: the columns *Communication Initiation* and *Information Storage*. Their sole purpose is to identify DVs. After identification, these columns can be hidden or deleted. The third column is *Induced-Configuration,* where the purpose is to find out whether the variable is connected to hardware setup or selection of application. In this sense, operational aspects are treated as application. These three columns might be related, and there may be some redundancy in this information. But this has to be confirmed by testing before suggesting changes. The last column, *Behavioural Types,* deals with what kind of behaviour a variable shows according to Kitamura definitions (Kitamura & Mizoguchi 2004a). The following questions provide a guideline for populating these columns, grouped according to the columns:

- Information Storage
    - Where is the information needed to populate this variable placed – is it inside the system boundary or outside?
    - Remember here that it is the "final" location of the knowledge, not the last part of the chain that is to be stated. For example, if we are to tell a controller that it is connected to a pump and the information needed is stamped on the pump, then the information is inside the system even if a human has to transfer it.
- Communication Initiation
    - When the communication is started, who starts it – the system or the outside user?
    - Again, using the pump to controller example, if auto detection is possible, the system will ask for pump info (hence system-started communication), whereas if no detection is available, the user has to tell the system that the pump is there and give its vital information (externally started communication).
- Induced-Configuration
    - Is the variable related to hardware setup or to application?
    - Presence of hardware, matching of values for range etc. count as hardware-related. Setting of variables connected with how the system is used would qualify as application-related.
- Behavioural Types
    - Which of the four base types does this variable belong to?
    - The purpose of this is to sort functional variables into basis and effect variables, or in the terms used earlier, into variables decomposed internally

(basis) and with external criterion (effects). These types also help to analyse the communication between the sub-systems needed.

As we stated in the beginning of this section, these columns are supplementary and only serve to create other information. In some cases, they can be omitted – for example, because of simplicity or when what is needed can be deduced directly. Let us now move on to the next step in making the EM model, the relationships.

## 7.4 RELATIONS GUIDELINES

The biggest step in making the EM model is to walk through all the relations and place them in the matrixes. With no IT support, we have to do this manually, relation after relation. How the relational information is presented is not important. We could have an Excel sheet with them all, or a stack of CRC cards where every module (and/or component) has its described rules or other form of documentation. Where we obtain and store the relation data is not so relevant, but all relations should be put in the matrixes. Let us look at the matrix construction and the population of them with relations.

### 7.4.1 CONSTRUCTING THE MATRIXES

When the initial PVM tree is complete, the internal and external matrices are created. For the internal matrices, the diagonal is empty and a diagonal-line is added in the empty diagonal from top right to bottom left. For the external matrixes, it is optional to draw a jagged diagonal (under the boxes to allow readability), here from top left to bottom right. This is done to show whether the matrix has been collapsed, see Figure 141. Note that internal matrixes are intra-domain matrixes with the same elements in rows and columns. Internal matrixes have left rotation, meaning the top row becomes the rightmost column. The external matrixes, on the other hand, are inter-domain matrixes with different rows and columns; here, the columns come from the abstraction level above. The external matrixes also have left rotation, making the top row of the higher abstraction the rightmost column.

### 7.4.2 ADDING RELATIONS

Once the actual matrixes are in place, we can start to populate them. The purpose is to show all relations in the matrixes, so if we encounter some logic that is not stated in our data but can be inferred, we should add it as well. As a guide, the researchers have made a list of possible relations. It is by no means complete, but in working through the case, we found no relations that could not be described with the suggested relations with a little reformulation of wording. There is a "black-box" relation called *calculations* for complex formulas. These then have to be described elsewhere.

Once the full PVM is generated and the matrices are formed and populated, the domain experts can be brought in. The model is then adjusted with their help.

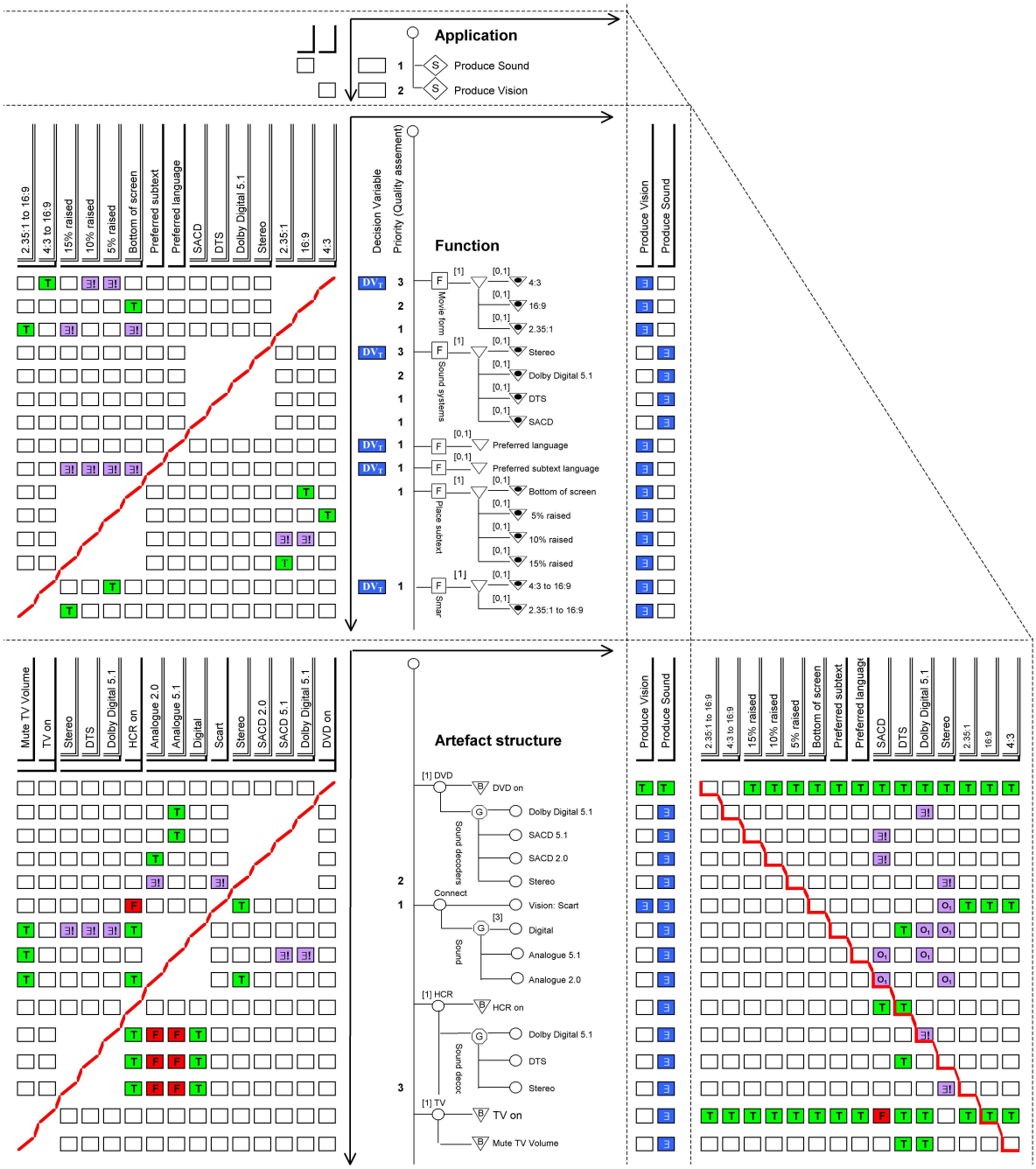



**Figure 141 – EM model for HES with diagonals**

This approach ensures that any mistakes and omissions can be corrected, and the initial model can be completed, without requiring domain experts for the entire modelling process. The domain experts' assistance is especially important in relation to formulating inferred relations. It is worth noting that there is a difference between modelling relations that lie within one EM and between EMs. Let us look at these.

### 7.4.3 INTRAMODEL RELATIONS

When dealing with a single EM, we have to map each level in relation to itself, and then each level in relation to each other. The relation list is constructed for use in both cases, and no major differences are observed in the two types.

### 7.4.4 INTERMODEL RELATIONS

On the other hand, we have to be careful when looking at relations between models. Here, we are in tricky waters and have to adopt a holistic view. These relations involve communication among sub-systems and hence rely on SBM and CM models. We have to make the parametric diagram based on the SBM and then rely on naming conventions and languages to perform the communication.

## 7.5 TRIMMING GUIDELINES

Once the first iteration of the model making is done, i.e. the tree has been populated and the relations placed in the matrixes, the model will most likely be too big. In this section, we discuss methods to iteratively go through the model to manipulate the design and hopefully improve it. There are two elements in this, the tree itself and the relations matrixes. As the matrixes are partly controlled by the tree, it can be stated that diminishing the tree size will also reduce the matrixes. Thus, it is like working on two different "axes", the tree and the columns. The tree can be "pruned" in several ways: Space for variables (true/false wording, binaries, not using a separate line for a variable name, showing values or not) and introducing grouping (functions, relational etc.). Removing columns is done through "collapsing".

In this section, we first go through the methods for pruning the tree, and then explain how columns are collapsed. As mentioned earlier, there are some ways to make the tree (the PVM centrepiece) as small as possible without losing details. These tricks can be summarized as follows:

- Space for variable
- Names and groups not on separate lines
- Relational grouping
- Constructing wise grouping for detail "turn-off". Can be done with same level/branch, and the first or first two levels for groupings can be "reserved".

Let us look at these in turn.

### 7.5.1 REDUCING SPACE FOR VARIABLES

The matrices that result from populating the modelling with relations will presumably be so large that they would be difficult to survey, and they might contain unnecessary information. Therefore, the next step is to reduce the size of the matrices in order to make them more easily handled and thus assist the analysis work. Reducing the size of the matrices is achieved equally by reducing the amount of space used by certain

variables and by removing variables and columns that show no information. By removing empty columns, a matrix can be collapsed, showing a staggered diagonal. We discuss the collapse of columns in a later section. Here, we focus on the space needed to show variables and their values.

There are two main reduction methods in play here: the *notation* of a variable and the *showing of values*:

- Notation of the variables and their solution space is a source of great space saving. Two types of variable are relevant here, variables that have *value range* and variables that can be restated as binary *true-false statements*.
    - Value range is when a variable actual only has one value, but that value can lie within a specific range. This could include both values with a distinct value such as type of input function (e.g. "Level") or a colour (e.g. "red"), but much more often the variable has a limited value range (e.g. [0..10]). Value range variables are considered to have a single value, not a range, because this is what they have in any given state. The range is just an "input mask". For variables of this type, the value is not given a separate line in the PVM, as with variables having several values. The variable and the value is instead written in a single line as follows: "**Variable**: Value". The variable is written in bold. An example:

      **Sensor zero offset:** [0...10]
    - The true-false statements are variables that have two values and the notation of the variable can be reformulated to a binary statement. These variables can have either one or two meaningful states. Variables of this kind are replaced by statements using logical operators and formulated as yes-no questions, for example: **IsVariableTrue?** Question marks are optional. An example:

      **IsFoamDrainingOn:** [True/False]
- Showing values is the other space-saving operation at our disposal. When populating the matrixes, values are shown. After that, if not-relations are on the actual values, the values can be hidden.
    - Hide values of variables if no relations are on both internal and external matrixes. We should even consider hiding values if there are many variables with many values, and relations seem to be "shattered" all over. Then, we can start with hiding values, replace the relation icons with group markings, and analyse the pattern.
    - Show values when relations are present for specific values.

Another type of space reducing is the hierarchical structuring of names.

### 7.5.2 HIEARCHICAL STRUCTURE OF NAMES

When constructing the PVM centrepiece, there is a tendency to place names of variables on a special line and their values on new lines. The same applies when grouping variables. This causes the matrixes to explode in size. Putting the variable

name and its first value on the same line can save much space. The same applies with groupings.

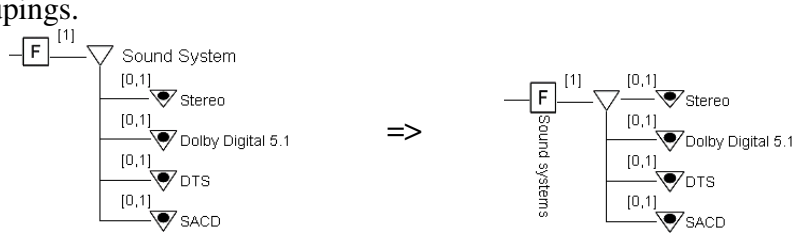



**Figure 142 - Hierarchical name hiding**

### 7.5.3 COLLAPSING COLUMNS

Collapsing empty columns does making the horizontal axis smaller. Collapsing columns can apply to "empty" columns, both when there are no relations, and when grouping has been performed. More on grouping later.

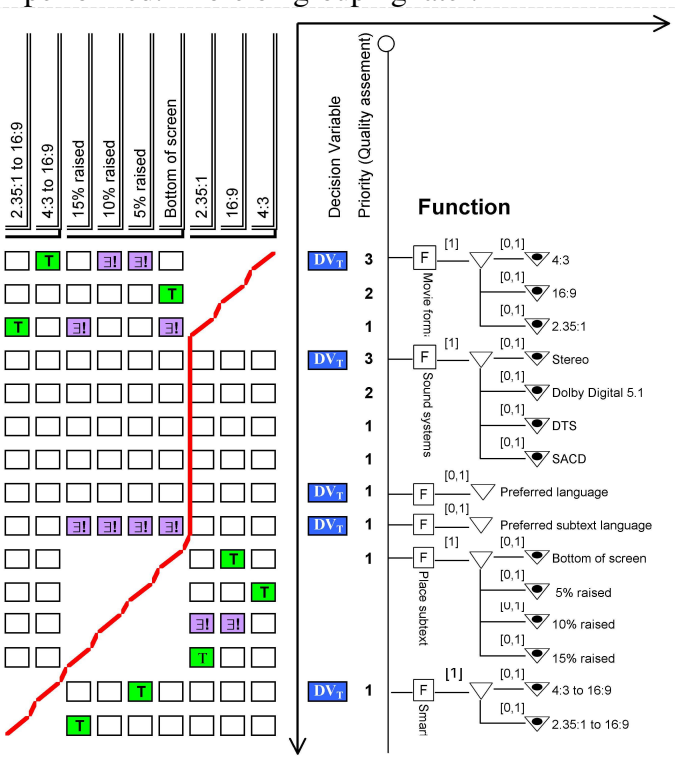



**Figure 143 - Collapsing empty columns**

In both the internal and external matrices, there are variables that have no causal effect on other variables, i.e. they have no relations to other variables, although other variables may have relations to them in terms of the IF-THEN structure. Therefore, there are several empty columns in the matrices that can be collapsed without hiding information, as shown in Figure 143. We can choose either to collapse all empty columns, or to keep columns that are empty but where the value in question belongs to a variable that has a relation for another value, i.e. when a variable has relations for only some of its values. In this case, the empty columns for the remaining values do not necessarily have to be collapsed. The reason for doing this could be purely practical. Completely removing all empty columns may require rewriting many of the variables so that the variable and the value with a relation are written in the same line and take up just one column of space.

It could also be interesting to still be able to see which variables have relations for just some of its values. In one of the student test cases, it was chosen to keep some of the empty columns for exactly these reasons. However, in most cases, it is best to collapse all empty columns because of the great need for size reduction.

**Figure 144 – Collapsed internal matrix with diagonal**

To signal the collapse of columns in the internal matrices, a diagonal can be used. By drawing a diagonal in the intersection between the variable and itself before collapsing columns, it is easy to see where a collapse has occurred. After collapsing the columns, a staggered diagonal is clearly visible. An example is seen in Figure 144. Notice that there are still a few empty columns present. The main difference in collapsing the internal and mapping matrixes is two-fold. First, the mapping matrix is most likely not square (if it is, it is by pure chance), and second, the rows and columns are not the same in the mapping matrixes and hence a diagonal has less purpose there.

The suggested way to visualize the collapse in the mapping matrixes is to draw a "jagged" diagonal with reversed inclination. The diagonal still indicates the collapsed columns as shown in Figure 145.

### 7.5.4 GROUPING THROUGH RELATIONS

With the first EM in place, we can use the relations to group the variables. There are two different types of groups:

- Groups where the variables only have relations to the other variables within the group
- Groups where there are also relations to variables outside the group. In this case, the group is characterized by having most of its relations within the group.

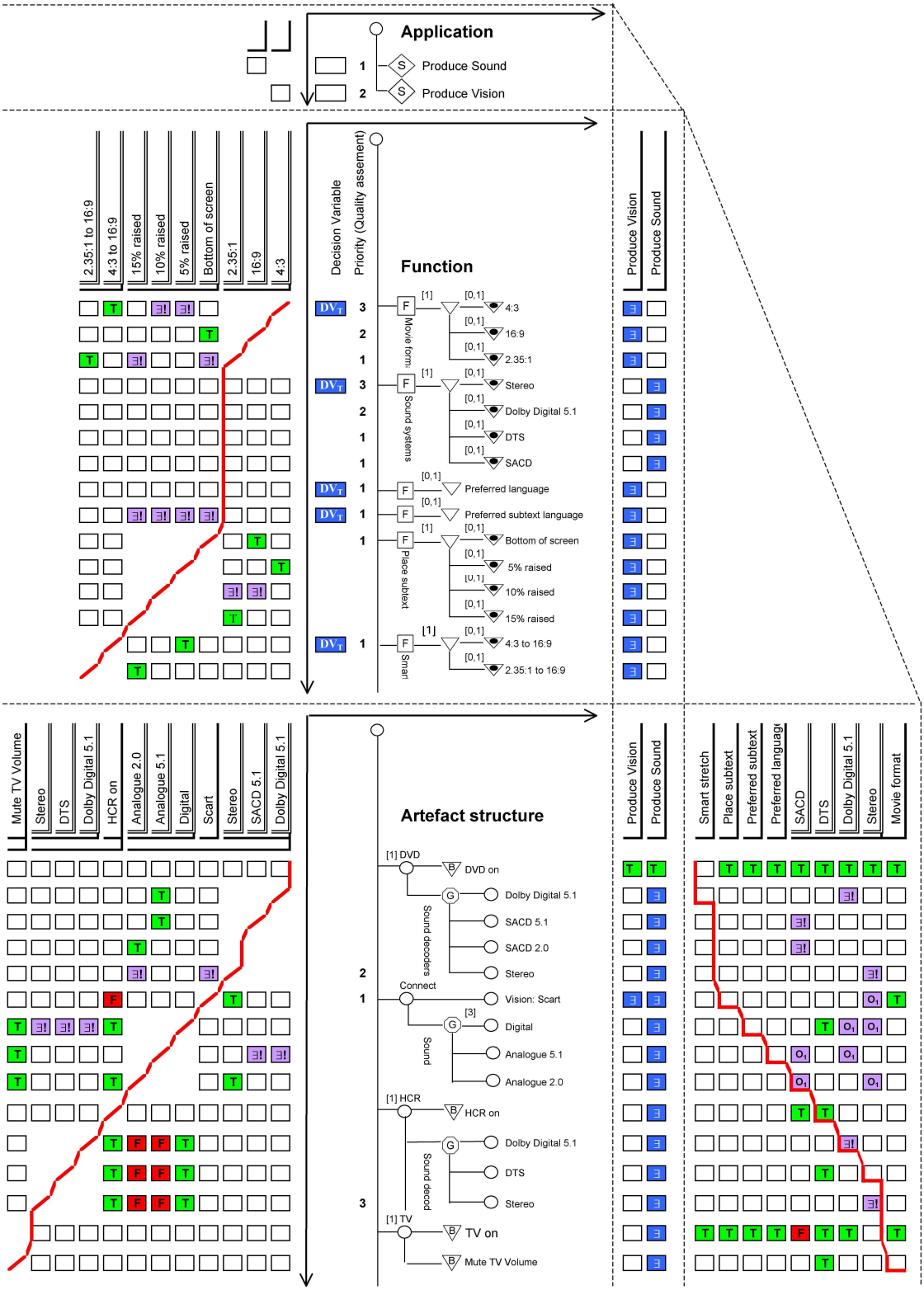



**Figure 145 - Collapsing mapping matrixes**

The best effect is achieved when grouping variables that only have relations to the other variables within the group or type 1 group. This gives the best demarcation of the variables/group and makes it easier to manage the variables and the relations. Variables are grouped by rearranging columns and rows in the matrices according to the mentioned requirements for groups. A computer can be used to do this. A formed group

is then designated a single group name, and under certain conditions it can be replaced by the group name in the PVM, thus collapsing a part of the matrix.

There is a slight difference in grouping the internal matrix and the external one. We can actually state that grouping should only be done on the internal matrix while observing the corresponding external one. The mapping matrix would of course reduce in size along with the internal on, at least with respect to rows. Grouping in the internal matrix can be said to have three variants:

- First, if the formed group has no relations to other variables outside the group, then the group can be replaced in the PVM structure by the group name in both the IF and THEN direction, collapsing both the columns and rows of the involved group variables.
- If the group has any relations to variables outside the group, then the variables can only be replaced by the group name in the IF or the THEN direction of the PVM where there is no relation outside the group in that given direction. Of course, it could be decided to collapse the PVM in both directions, but this would require that the relations are not detailed; instead, only an indication of the presence of a relation can be shown.

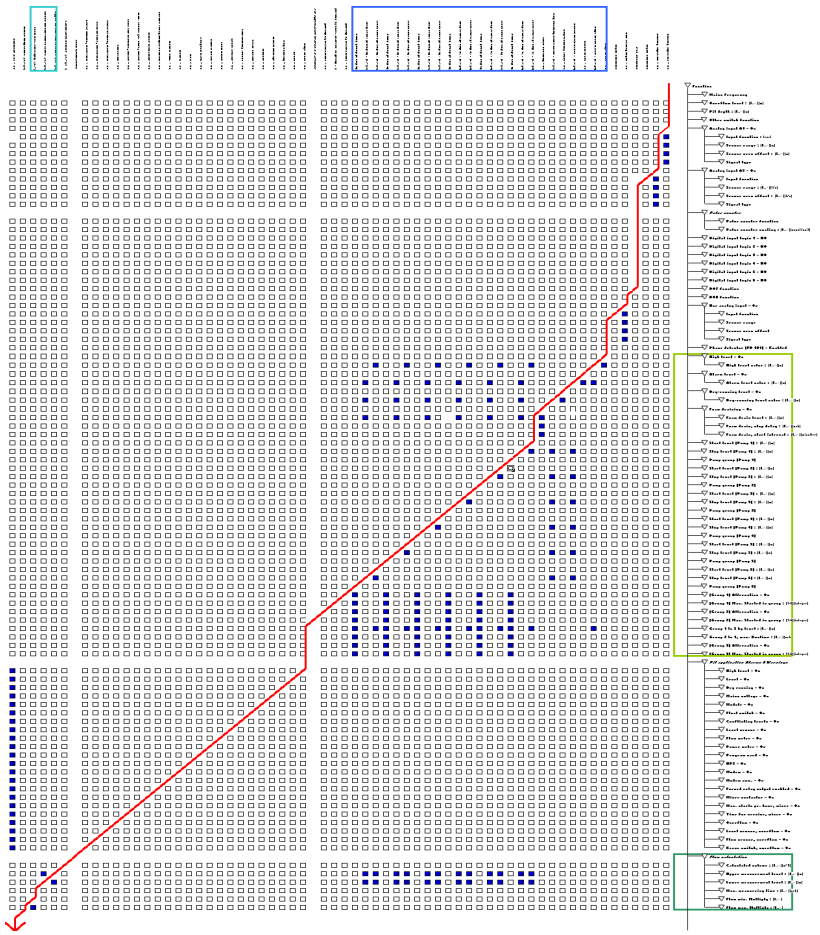


**Figure 146 - Variables suggested for grouping based on relations**

We can state that as a general rule a group of variables should not be replaced by the group name in either the IF or THEN direction when values are shown, unless they

have absolutely no other relations to variables outside the group. The following set of figures (Figure 146 to Figure 148) show an example of a grouping process.

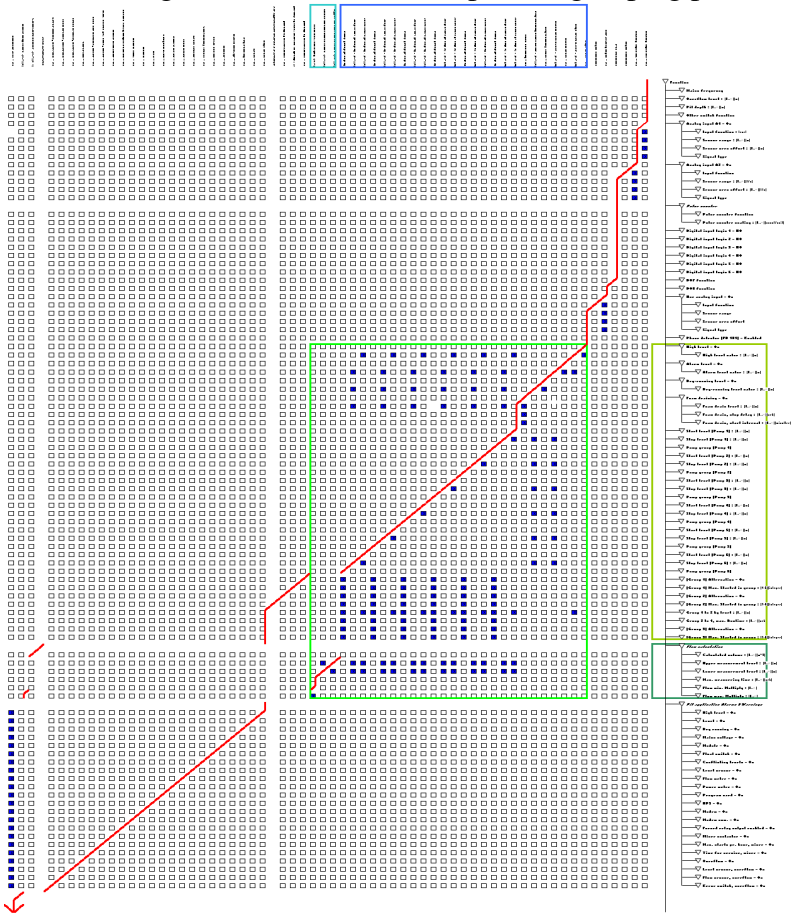


**Figure 147 - Variables moved both in rows and columns to lie together**

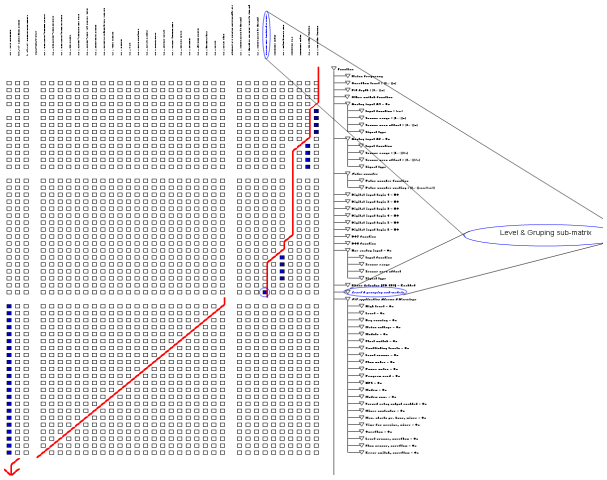



**Figure 148 - New group name given**

The software used to model the PVM should be able to shift between only showing the group name and showing the full group and the variables with values contained within. A final note concerning grouping and hiding of variables is that it is fine to regroup variables for an internal matrix, but as stated earlier, it might not prove worth the effort to regroup variables in the external mapping matrix.

### 7.5.5 REDUCING THE DEGREE OF DETAIL

Even though the model can show all the details, it is most beneficial to be able to show less in certain cases. This can apply to strategy decisions, communication between domain expert groups and others alike. Details can be reduced mainly by two means: through the turning off *values* for variables and through *grouping* the same ones.

- Turning off values, or rather not showing variables' values in the tree. This option is like having a toggle button in the model for show versus hide values. Essentially, this assumes that the model is computerized and driven by software, not made manually. Just this simple action of toggling values requires a tremendous amount of work to do manually in Excel. An example is shown in Figure 149.

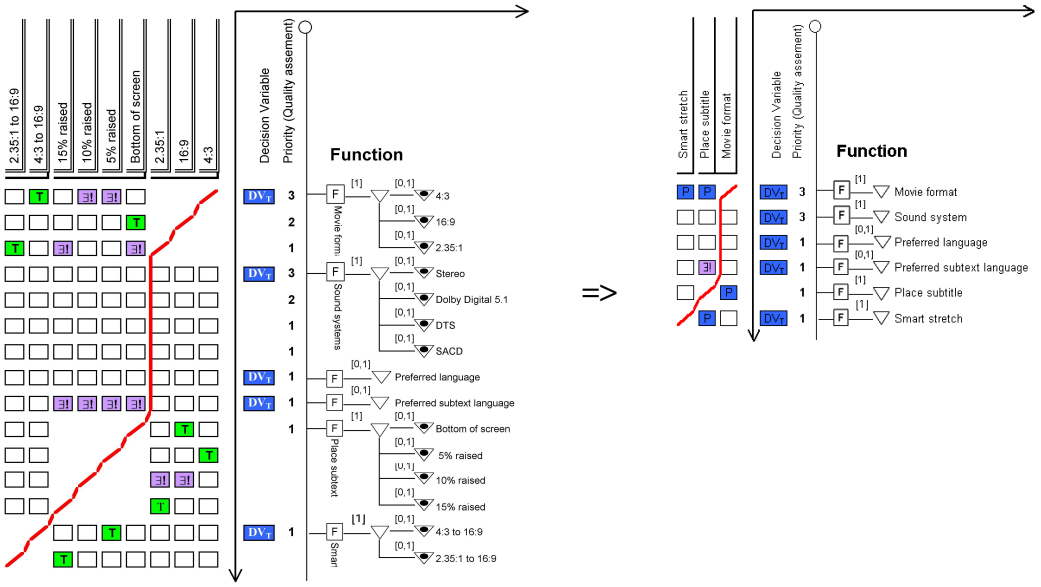



**Figure 149 - Hiding values in EM model**

- The other possibility is the grouping of variables. As discussed earlier, this can be done by reserving some levels of each branch for grouping. By grouping, the lower levels of the branch can be "turned off" and thus collapse the tree considerably. Toggle functionality would also be very beneficial here, as it can allow some analytical work to be done. Grouping present itself just like hiding values (see Figure 149).

The beauty of the model is that no matter what detail level is chosen, the model has the same overall look.

## 7.6 ANALYSING THE MODEL

Having completed the model, it is time to analyse it and try to draw some conclusions. Analysing the information in the matrices can be done by identifying patterns, analysing the collapsed diagonal, evaluating the prevalence of certain relations, and finding groups of variables that are either uncoupled to other variables outside the group, or mostly related to variables within the group. There are six main issues that should be investigated:

- Voids in the matrix

- The shape of the diagonals
- The distribution of relations within a matrix and the ratio of relations between internal and external matrices
- The types of relations present - which relations are most frequent, i.e. the colour palette
- Variables that have no relations to other variables, i.e. variables that have both empty rows and columns
- The decision variables

Now, let us look at these in turn.

### 7.6.1 VOIDS IN THE MATRIX

When visually inspecting the matrixes, we notice voids in it. The voids have meaning; they stand for "no relation possible". By default the diagonal in the internal matrix is void, since a variable (or its value) cannot have a relation to itself, as shown in Figure 150.

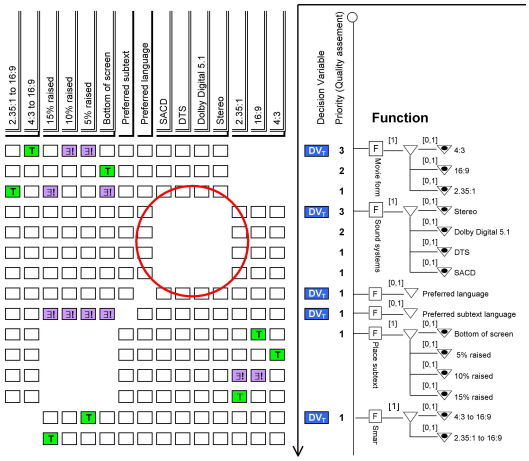



**Figure 150 - Voids in the matrix, variable to variable**

Lines, columns or both can also be void. This means that the corresponding item, probably a group name or variable name, cannot have defined relations. This is shown in Figure 151. We can intentionally place such names in the matrix to create "air" in the matrixes and hence increase readability. Too many voids point to an inefficient tree structure that should be trimmed.

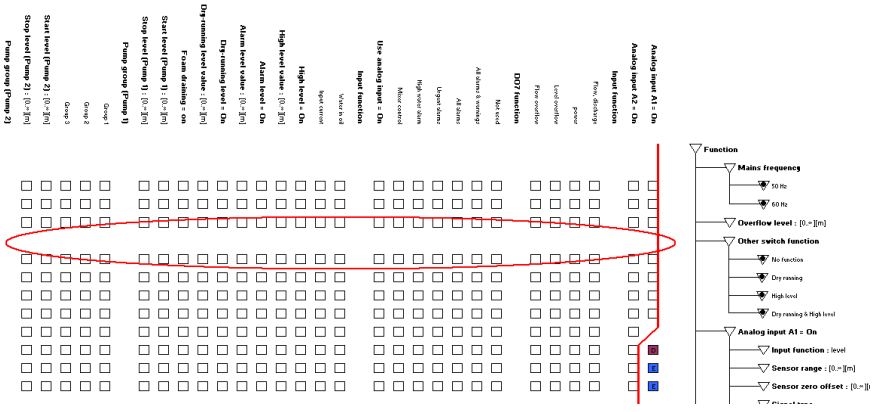



**Figure 151 - Voids caused by grouping and variable names**

The diagonal is the next thing to look at.

### 7.6.2 DIAGONALS IN THE MATRIXES

The diagonal in the internal matrix is an excellent tool for seeing how coupled variables are. If the diagonal is very collapsed, it usually points to an uncoupled variable structure, as seen in Figure 152, though we have to look at the mapping matrixes as well as cascading effect, as this could be the way the product is structured. The mapping matrixes can also be collapsed; then, an inversed diagonal is helpful in visualizing the collapse. Note that the mapping matrix will not have a void diagonal either way, as it maps between levels, and all items in each level can relate to all.

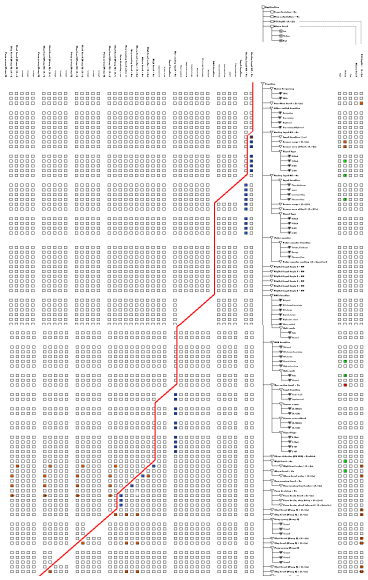


**Figure 152 – Collapsed diagonals and relations in internal matrix**

Because of this, the jagged diagonal (as seen in Figure 153) can be a little disturbing, but it is considered necessary to show in a collapsed matrix.

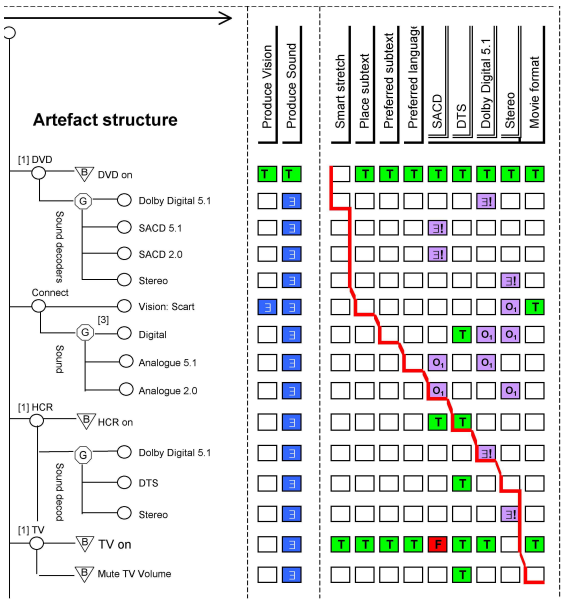



**Figure 153 - Collapsed mapping matrix**

A more general issue is where the diagonal is a sub-part of the pattern recognition of the whole model. This is the subject of the next section.

### 7.6.3 PATTERN RECOGNITION

Observing the pattern visible in matrixes serves three main purposes: to identify grouping possibilities, to create cascading effect through adding relations, and finally to identify current and future decision variables. Explaining these patterns is hard and an example is in order. Some statistical numbers can perhaps assist in comparing matrixes, e.g. evaluation through the "fill-in" percents. We can also develop a spread number to find how "clustered" the matrix is. This is the same action as suggested with DSM, though the purpose here is usually to generate groups and decision variables, not sequential actions as in the DSM (Steward 1981).

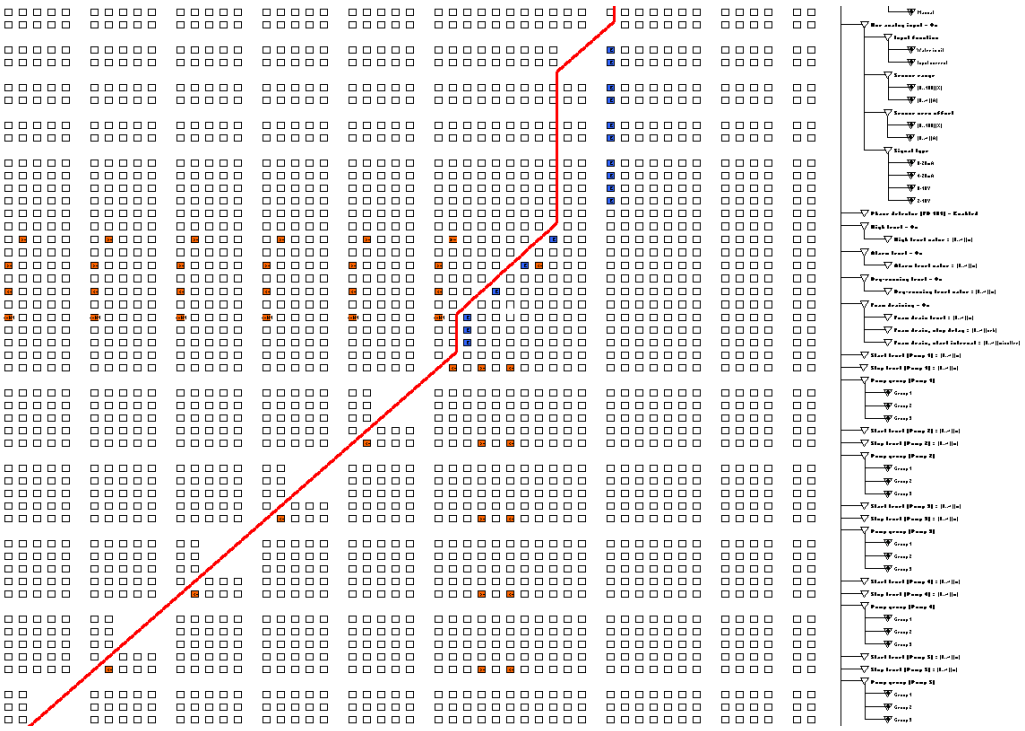


**Figure 154 - Patterns in the matrixes**

Think of these patterns as filled versus not filled, not in terms of exactly what kind of relations the markings stand for. For this, we use the colour palette.

### 7.6.4 COLOUR PALETTE

A special type of pattern recognition is the "colour palette". This is used to visualize the type of relations present in the matrixes, especially when a large number of "Exist" relations occur and there is a limited cascade effect, which then requires the user to set a large number of variables manually. Another attribute of the colour palette is its compact form. Here, we have removed all the intermediate columns and rows to form a true mosaic. In this way, the colour palette is one-fourth in size compared to the original model (1/2 in each direction).

It is important to recognize that the colour palette is thought to be an overview tool and this can cause some details to get lost. Yet another issue to analyse in the model is how connected variables are. More precisely, when looking at the individual variables, it is necessary to be aware of whether the variable is "stranded".

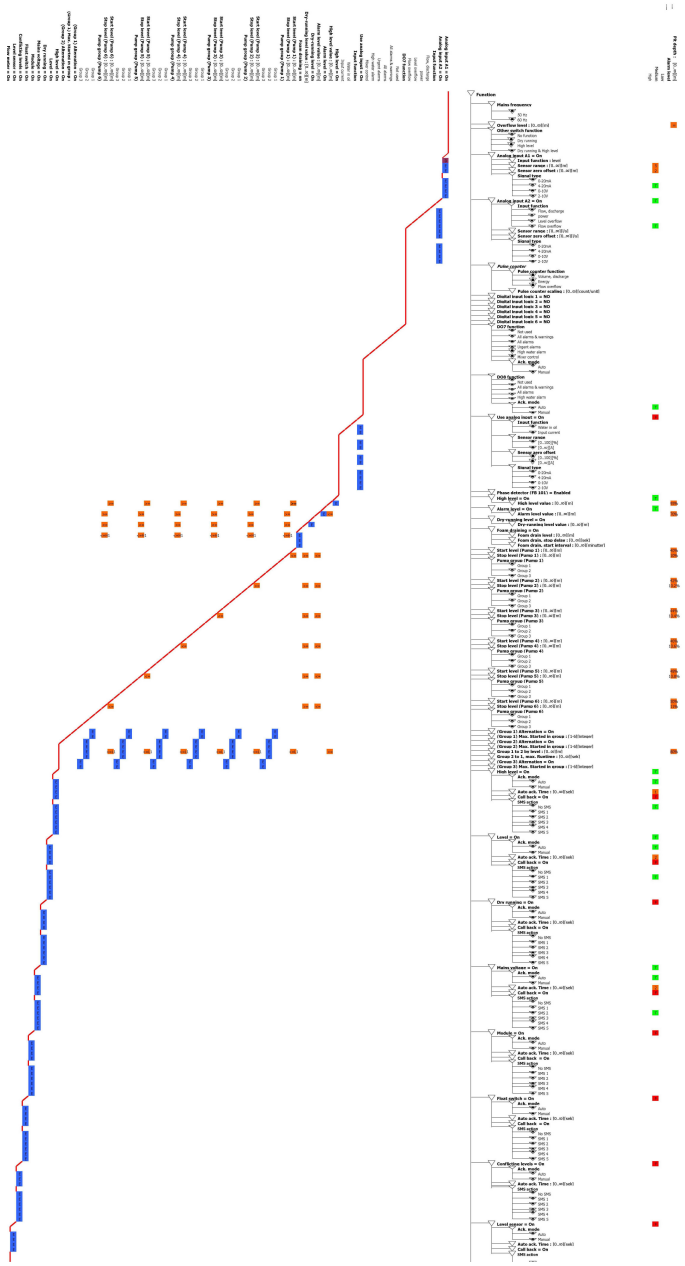


**Figure 155 - The colour palette**

### 7.6.5 STRANDED VARIABLES

Variables that have no relations to other variables, i.e. variables that have both empty rows and columns in both internal and external matrixes, are hereafter called "stranded", based on the researchers' belief that a variable that is not connected to anything should be considered "strange" and therefore examined. There can of course be instances where this is normal, but they should be checked. The same logic applies to a single value of a variable – if it is stranded, it should be examined.

Finding a stranded variable is actually very difficult to do manually. In the dataset the researchers worked with, many were found, and it was only because of their share number that they were easy to find. In models where only few exist, it could be tedious. This is an aspect to remember when constructing IT support for the EM model.

Stranded variables can indicate missing relations, implicitly stated relations, designer's intention is obvious, and user knows how to relate etc. The researchers believe that stranded variables are the first thing to scrutinize when trying to improve AS-IS and suggesting TO-BE. All this leads to the actual purpose of the whole method, to reduce the number of decision variables.

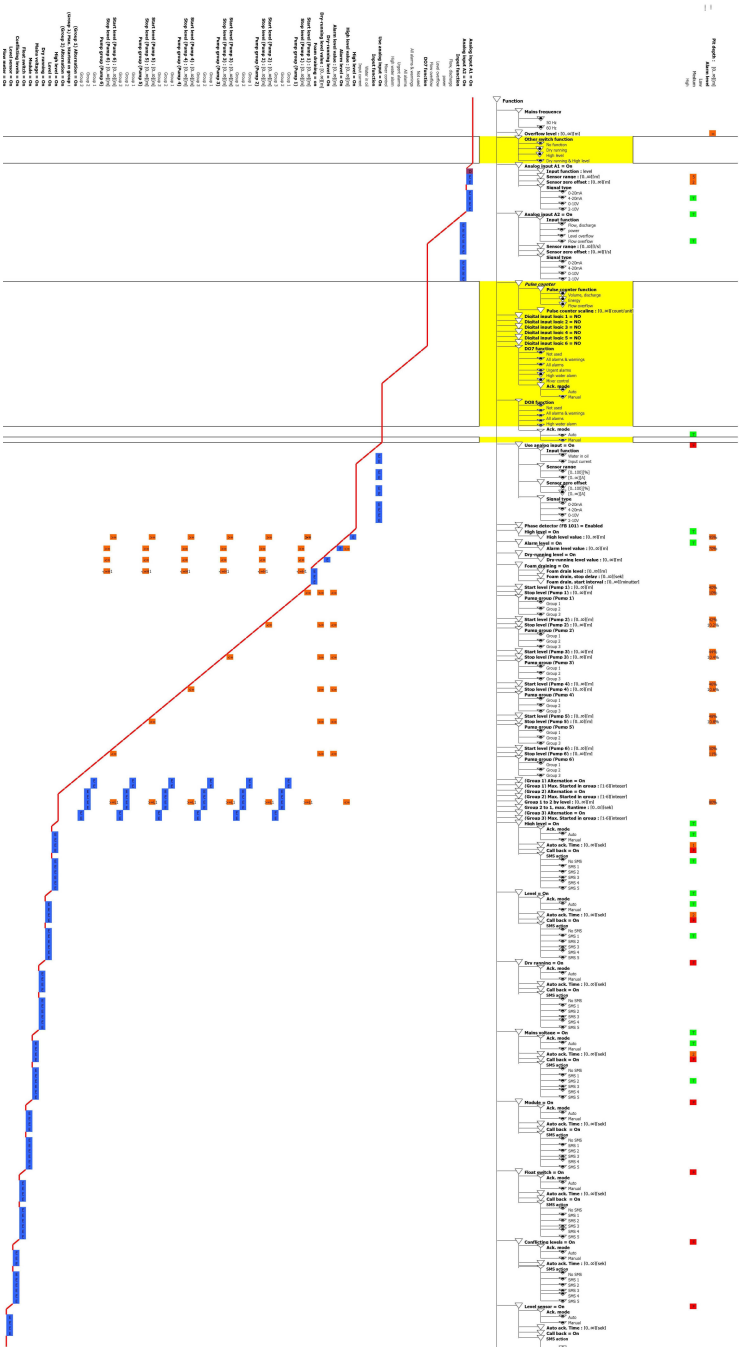


**Figure 156 - Stranded variables (highlighted in yellow in PVM)**

### 7.6.6 DECISION VARIABLES

Decision variable, a variable that breaks the system boundary and requires a user to take action, is one of the core concepts presented in this thesis. It is the goal to analyse the current product structure in order to suggest an improved structure where the number of DVs is drastically reduced through cascading effect and relations. Most importantly, the analysis should lead to the definition of a limited number of decision variables that should lie in the application abstraction level. Here, we have three different types of DVs to play with. The three DVs are, as presented earlier, DV, tentative DV and internal DV. It is important to remember that in the end, it is only the DV we intend to reduce. We are not so concerned with the other two, as they mostly remain hidden from the end user.

This concludes our walkthrough of making and analysing the models suggested in this thesis. We have presented the models and a technique for making them. We have tested different aspects of the models and noted what could be done better, along with ideas for further research. The whole process is extensively based on theory. We have presented our rationale for this work, why it should be solved as we suggest, and what we expect it to give us. Let us now conclude this thesis with a summary of contributions and a discussion of several issues.

# *Chapter 8*

# DISCUSSION AND CONCLUSION

This research has been motivated by the need to simplify installations of such complex product systems as water supply systems. It assumes that several parameterized sub-systems have to be connected to form an overall system. As parameters imply software, this is indeed kind of related to software engineering and can be seen as such. A sub-goal is to suggest a method that is tightly coupled with software implementation and does not require double work in modelling and programming. The thinking presented here is also very related to distributed artificial intelligence and can be viewed as a tool for constructing systems and even a start for developing a method that has been sought after (Wooldridge & Jennings 1995). Another thing worth mentioning is that benefits from the suggested solution can only be fully realized at system level, meaning that initial costs of knowledge engineering may be high. It is only when looking at the complete life-cycle of the product system that the rationale makes sense.

This chapter is about all the work that has been done until now and is an attempt to put it in perspective. The researchers try here to throw light on the key elements of the analysis and build them into a viable solution. The chapter is structured as follows: summary of contributions, answers to research questions, discussion on issues identified in the process, possible impact, a crack at theory creation, and finally, conclusion with recommendations for the future.

## 8.1 CONTRIBUTION OF THIS THESIS

In this thesis, we have presented a novel way to model products with the purpose of improving after-sales service. Let us go through the contributions made here and summarize them.

On the general level, this thesis deals with two aspects that could be considered new:

- The use of configuration in after-sale services.
- The modelling of product knowledge when dealing with unknown solution spaces. The difference here can also be highlighted as the difference between sales configuration and embedded configuration for after-sales services.

These general aspects can then be supported with the actual contribution made in this thesis. The contributions can be grouped into modelling efforts and literary support. The modelling contributions include:

- A modelling technique for modelling knowledge that is to be used in configuration of after-sales service support tasks.
- Encapsulation model with strict relational usage, predefined relations, visual presentation of complexity, and identification of decision variables.
- Visualization of software variables and their relations.

- Modelling technique that utilizes our human capacity for visual analysis and takes into account our limitations.
- Supporting models and processes to make encapsulation models.

The literary support that has served as inspiration and concept generation for the modelling technique includes such contributions as the following:

- Analysis of relations for use in product modelling.
- Analysis of decompositions for use in product modelling.
- Analysis of functions for use in product modelling.
- Analysis of artefacts for use in product modelling.
- Analysis and concept inspiration from system theory for use in product modelling for configuration.
- Analysis and concept inspiration from Mereology for use in product modelling for configuration.
- Analysis and concept inspiration from engineering design for use in product modelling for configuration.
- Analysis and concept inspiration from knowledge engineering for use in product modelling for configuration.

In the opening stages of this research, we asked some research questions. Let us return to them and answer them in turn.

## 8.2 ANSWERING RESEARCH QUESTIONS

As our next step in this discussion, let us walk through the research questions and try to answer each of them.

### 8.2.1 CONCEPTUALIZING

The first research question was how to construct a concept to use for configuration embedded in each sub-system. The formulation was:

> *How can product knowledge be modularized in order to allow encapsulation but without losing overall system integrity?*

The short answer is:

> The introduction of abstraction levels in describing the product as suggested in many methods is also to be used here. What is to be done differently is the encapsulation of the product data in the form of a single model placed inside each sub-system. The functional description of the product leads the way to facilitating communication and reuse. A standardization of the functional descriptions is also necessary. All of this takes place in a single encapsulation model supplemented with communication ontology.

As a supplement to the first research question, several supporting questions were formulated:

1. *Conceptualizing embedded configuration*
   - *1.1. What is the role of modularization in embedded configuration?*

     Here, modularization relates to the sub-system structure. Modularization is essential to embedded configuration. It can be stated that it is a prerequisite,

with all the things that follow such as standard interfaces and preferably functional one-to-one mapping.

*1.2. How does one encapsulate product information?*

There are several factors here. We have to introduce three abstraction levels: application, function and artefact. Application states the context; function describes the inner workings; and artefacts list the components present. Between these levels, we have to map each element to both elements in the same level through internal matrixes, and to elements in other levels through mapping matrixes. We have to define decision variables – what variables need inputs from "outside" the model, both from other models in the system and from outside the system boundary. The most important aspect here is to maximize the "cascading effect" within the model so decision variables are kept at the lowest number possible. In other words, this is about capturing design knowledge and incorporating it into the models.

*1.3. How do modules interface each other?*

We assume known standard physical interfaces. The interfacing should thus be on the meaning level.

*1.4. How do they communicate?*

Again, as in the previous question, communication is on the meaning level. By introducing standard vocabulary for all three abstraction levels, ontology for sharing and allowing negotiation between sub-systems should be meaningful communication.

*1.5. Is it necessary to divide communication into layers?*

Yes, it is. Communication should be hierarchical, meaning that sub-systems should start at application level and only proceed to next level (functions) if necessary. The same goes for moving from function to artefact level. This is in order to adhere to Suh's second axiom (Suh 1990) on minimizing the information content; here, this relates to information needed in communication messages.

*1.6. Who controls in a system of peers and how?*

The models are made in such a way that they support both a system of peers and hierarchical control. This is done by giving services and function a quality evaluation. Hierarchical control is achieved with different quality levels on the same functionality/service, and a system of peers appears when quality levels are the same. This then triggers negotiation to determine working conditions.

*1.7. How are new versions/upgrades of modules handled?*

As all components and sub-systems are described in functional capacity and put in context via the application level, a change in component or sub-system replaces known functionality and hence does not influence the overall system. This of course assumes standard interfaces and is achieved through layered communication.

*1.8. How does one ensure information flow (no redundancy)?*

To reduce data entry to a bare minimum, the following is suggested: make functional description with standard language of the artefact, embed product data into the sub-system, allow communication between sub-systems and a negotiation between them for determination of data flow and control.

The second research question is on modelling.

### 8.2.2 MODELLING

The second research question was on how to model embedded configuration. The formulation was:

*What should the modelling technique be like in order to support modularized product knowledge?*

An answer to this question is briefly:

The technique must incorporate three aspects: the system breakdown model, the communication model and the encapsulation model. The SBM depicts the overall system and points to the units to be described in EM models. The CM helps identify what is needed in information sharing. The heart of the technique is the EM model. It describes the sub-system in detail, with all relations in place. It is constructed with three abstraction levels with stringent relation mappings in the form of matrixes.

There are also some supporting questions that are worth looking into and answering in turn.

2. *Modelling product knowledge to support embedded configuration*
   - 2.1. *How does one encapsulate variance (hiding internal parameters)?*
     There are two kinds of encapsulations, within a sub-system and within the system. The first is to "hide" as much variance of the sub-system as possible from the whole system; only communicate what is necessary, meaning here that communication is through internal DVs, which are hopefully at least on functional level. The second is to ensure internal information flow within the system, here mainly in hardware- and application-induced configuration.

   - 2.2. *How does one tie functions and structure together?*
     Like all other relations within a sub-system, functions and artefact structure are connected through the design structure matrixes (DSM), here called mapping matrixes. By introducing a standard set of symbols, all relations can be depicted in these matrixes.

   - 2.3. *How is the model interfaced?*
     Encapsulation with clear interfaces is the goal here. The interface is the decision variable (DV). They should be constructed with care, described with standard language (an ontology), and on a higher level – on service or functional level. Communication should be driven by DV and ontology.

   - 2.4. *How does communication function between PVMs?*
     As in the interfaces, communication between sub-systems is through ontology that shows how elements connect, what is needed and how to go about communicating.

   - 2.5. *What kind of complexity is needed in the model?*
     All parameters have to be related and placed. There may be relations (rules) that are too complex to show in the matrixes; they should be described elsewhere. It is however the goal to try to have as many relations as possible in the model. The whole complexity issue is a big deal and much effort has

been spent on dealing with it. A toggle between details should be allowed and is in fact necessary, because of model size.

*2.6. How is dynamic communication between different PVMs allowed?*
A twist on communication is dynamic communication. Here, we refer to things that happen when system attributes change without user input, e.g. if some sub-system breaks down. By allowing negotiation and giving the system a measure of quality, it is hopefully possible to gain some autonomy. This is achieved through quality evaluation of services and means of finding the best available setup.

*2.7. Where is product data stored?*
The main issue here is not to rely on an external IT system, but to store product data inside each sub-system in order to create encapsulated storage for product data. This is by no means the only possible way to do it, but it is selected, because the researchers feel it is the only viable long-term way of making an intelligent system setup.

*2.8. Where are rules of combination stored?*
In an ontology, holographic in nature (all sub-systems have basic knowledge on what is needed), so that each sub-system can tell what it needs and is able to supply to the overall system.

*2.9. Where are rules implemented?*
All rules are implemented in the matrixes. The rules restricted to one sub-system are mapped in both internal and mapping matrixes. The combination rules, rules between sub-systems, require "dummies" (phantom variables) to allow rules to be visualized in matrixes.

The third research question is on processing, more precisely the process of making the models.

### 8.2.3 PROCESSING

The third research question is about the process for making the model. The formulation was:

*What should the process for building an embedded configuration system be like?*

The brief answer to this could be:

To construct the core model, the encapsulation model, both for the individual sub-system and the stacked EM for the whole system, we have to go through several iterations. First, we have to draw the system boundary, what is inside and what is not. This is achieved with the SBM. The system is then described with EMs. What is of most interest here is the inputs that break the system boundary. These decision variables are the core issue when making embedded configuration. The whole purpose is to reduce DVs to a very reasonable size.

Supporting questions to the third research question and their answers are as follows:

3. *Developing embedded configuration system*
   *3.1. How does one decompose an embedded configuration system?*

The suggestion here is to work from the existing product assortment. From this, identify what we call archetypes, a general artefact that represents a group of products. For the whole assortment, create function streams, a description of what is to be achieved on a very high functional level. The last step is to overlay the archetypes and the functional streams to create the system breakdown model (SBM).

*3.2. How does one make the models?*
Starting with SBM as the initial suggestion for internal decomposition in the encapsulation model (EM) and then iterate with decomposition guidelines. The model making relies on standard languages for facilitating communication and reuse.

*3.3. How does one relate the models to the environment?*
The SBM shows the system boundary, and the context should be clarified in services and applications. Decision variables are the main relation between system and environment.

*3.4. How does one acquire data to populate the models?*
We have assumed that data is accessible and the structure is sufficient. In this case, we have used all documented information on the products, along with software tools and expert interviews. Knowledge acquisition is not the focus here, but it provides material for further research.

*3.5. What are the success factors of embedded configuration?*
The most relevant factor is the number of decision variables. When comparing AS-IS to TO-BE situations, a serious reduction of DVs should be registered. The major success factor is hence the DV.

And the last research question asks the big why: why should this be done, in short, what is the rationale for all of this.

### 8.2.4 RATIONALE

To the discussion of the fourth research question:

*Why should modularized product knowledge and embedded configuration systems be implemented?*

The answer could be, in short:

There are two main reasons for implementation, and they are not mutually exclusive. The first is the reduction of complexity, and the second is saving money through reduction of resource usage. Along the way, several other reasons can also be convincing. Complexity is present in many aspects: product complexity, installation complexity, operational complexity etc. The focus here has been on installation and hence decision variables. To achieve complexity reduction, we have to map the product data and construct the models for visualization of the complexity. On this basis, we suggest new relations to create sensible decision variables. The aim is to reduce decision variables drastically. However, we must also aim to maintain the flexibility of the product. The second reason is cost reduction. When decisions are to be made on many less items, the time spent is

reduced. This means less resources and hence reduced cost. This aspect has not been proven yet, as it requires either an advanced experiment or actual implementation.

This concludes the answers to the research questions. The process has however not been without problems. Or rather, there are still several issues that need attention. Let us look at some of them and see if we can shed some light on how to proceed.

## 8.3 DISCUSSION OF ISSUES

This section is about issues: issues that have risen during the work, issues that can be derived from the results, and issues that can lay the ground for future research.

### 8.3.1 RELATIONS

The most important aspect of this thesis is relation. By introducing system theory, we imply that focus is on Systemhood and not on Thinghood. As discussed in the theory chapter 4.1, Systemhood focuses on relations, and this is the focus we want when conceiving of embedded configuration. This also brings forth some aspects here that are worth mentioning.

- People try to make the relations as simple as possible! This leads to an uncoupled design and could explain why the relation list is so empty. Only fractions of relations are considered relations: things like decomposing, grouping, sorting and the like are usually not viewed as relations.
- Relation definition is another issue. When working through theory, it generates some very detailed categories of relations. Trying to map them to the relations actually used is very hard. Rooted in the earlier point, people try to isolate things when possible and keep relations as simple as possible. This results in a very simple set of relations to be used in the matrixes. And this set resembles only a little the theoretical categories identified through the literature.
- Another practical categorization of relations is presented by (Li, Raskin, & Ramani 2007). They give 12 relation types (see Table 21 on page 91), all of which can be represented in the method suggested here. What is interesting is that some of these relations are represented with placements in the current technique, e.g. *has_function* just places *exists* (or another type) in the mapping matrix F-S. The same goes for other types.
- Creating cascading effects in the encapsulation model through relations is hard. None of the products analysed showed any measure of cascading effect but were very flat in their structure. This is probably because of lack of overview. When 1300 parameters are introduced, it is enormously difficult mentally to relate them to any suitable degree. If we, as designer, cannot visualize them and their relations, we will probably try to keep things as isolated as possible. But this just causes problems for the one that has to set all those parameters.
- Making applications is rooted in creating cascading effects. If relating is difficult, applications cannot be made. Applications are also about capturing intentions of the design and relating them to the actual customer need. This is something designers are not used to stating explicitly, at least not in detail. And there is little (or no)

tradition for keeping track of such rationale downstream. In the method suggested here, making applications is probably going to be the hardest step.

- One can wonder about relations and their types. Why make the internal relational matrixes and not just put all those relations in the mapping matrixes? This is of course possible, and there are without doubt some cases where it is appropriate. However, we would like to suggest that the functional descriptions are kept, because we believe they are the foundation for streamlining and reusing functionality within product systems. Our rationale in this case is in line with arguments for use of modularity to gain flexibility and maintain costs at a reasonable level. This is a topic worthy of future research.

### 8.3.2 VIEWPOINTS

Much of the tension experienced when testing the technique was caused by differences in worldview. The field of software engineering has a fundamentally different view of things than the field of engineering design. Most work in SE is logic-based and does not rely too much on our visual capabilities. Even though SE uses visual models like UML, these visual efforts are "logic pictures" and do not rely on the human visual processing capability, since these pictures need to be interpreted into logic. ED, on the other hand, is much more "inspirationally" driven, which generally means a greater use of our senses.

Since this method aims to merge the two fields a little and hopefully bring the strengths of each to the other – i.e. the logical strength of SE combined with the purposeful overall aspect of ED – it would be very interesting to look at the viewpoint difference in more detail. This is not to criticize either field, just to say that both would benefit from more synergy.

### 8.3.3 EFFECTIVE VS. EFFICIENT

This work has been about making things work. It is about making the method effective. The researchers have no doubts, as such, that the method is effective. There are some concerns about its efficiency, but the researchers believe this to be solvable through implementation of an IT-support tool. Much of the inefficiency is rooted in manual work in manipulating the models, which can be helped by easing the manual work with computerized support.

Another type of inefficiency can be said to exist within the method, and this could be the implementation into a running KBS. As no extended prototype has been constructed, this cannot be tested or even speculated upon.

### 8.3.4 COMPLEXITY

A core issue here: what is complex and what is not, and equally, what is enough complexity reduction?

Several questions can be formulated for further digging into the complexity subject:

- When is a system complex enough so that Kefec would be applicable?

- Are there systems that are too simple for the method?
- Is it possible to suggest a "complexity measurement" for evaluating the two previous questions?
- Can PEAS be tied to such a measurement? At the moment, PEAS is not directly coupled to the task. Such a coupling would very probably be beneficial; it would also work as input to a measurement.
- Is reduction in DV adequate measurement for complexity reduction? How is it possible to check this?

### 8.3.5 INDUCED-CONFIGURATION DISCUSSION

*Induced-configuration* is introduced as an attempt to categorize DVs and relations for a quick measure of possible automation degree. As we have discussed, hardware-induced configuration (HIC) should become completely automatic (read: no DVs) while application-induced configuration (AIC) always needs some user input. What we are missing here is the degree of connection between AIC variables. Marking a variable as AIC does not point to how many DVs an AIC group will have. It is precisely this ratio we need. The question then becomes: What is needed to find the connectivity between AIC variables, probably across EMs?

To deal with this, we have introduced several evaluations on the PVM centrepiece. Introducing things like *function group*, *communication initiative* and *information storage* were attempts to rectify this issue. It is still open for debate if these are sufficient to solve the problem. In any case, a new term, some sort of "AIC-DV" should be devised.

Another issue in the induced department is whether a new *induced-configuration* term is needed, a sort of "*Operation-Induced-Configuration*" (OIC). This would mean splitting the AIC into two units, the selection of application and its operation. But since we have chosen largely not to consider operation in this iteration, it remains to be seen if OIC is necessary, or beneficial for that matter.

### 8.3.6 WHAT IS FINISHED?

The researchers realize that the job is not done. Most effort has gone into the EM model and the literature to support it. The CM part is lacking and requires much more work. Testing has only done on the knowledge representation level, but a prototype is required for full concept proof for the system interaction.

Even though the goal was to develop agent model technique, the journey is not completed. Focus has been on formalizing relations and creating a computer-friendly EM, and not so much on the rest. During the last moments, we have looked some at SBM and system breakdown, but the CM is largely unexplored, or at least not fully tested. Much work has gone into laying the background, searching the literature for useful bits and pieces, and analysing the actual problem with the purpose of generalizing it. So, this work is highly conceptual, a grounded theory that hopefully is also very practical.

## 8.4 IMPACT – WHERE COULD IT BE USED?

This thesis has focused on conceptualizing a formalized technique for a specific task, embedded configuration. Most effort was used on scoping a wide domain for building blocks. The resulting method is not limited to the initial task. Here are some top of the head speculations on where Kefec could apply! The technique could be used in the following situations:

- *Knowledge Engineering,* in general. When constructing KBS, a model with such visual capacity as presented here would most likely be very valuable. Highlighting complexity and visualizing relations are relevant here.
- *Systems Engineering,* when evaluating purpose and system decomposition. The method could serve as a guideline for making customer-need guided decomposition, an abstraction-led way to system breakdown.
- *Artificial Intelligence*, the construction of Immobot systems. The modelling of each immobot could be done with an EM.
- *Engineering Design,* when designing mechatronic products to create the link between the mechanics and the software, the relations in EM. The EM could also serve as a visualization technique for rationale and for documentation of design intentions.
- *Software Engineering,* for creating overview of parameters, a guideline for making user interface by letting the abstraction lead the way, and finally for mapping relations.

The usage of Kefec in these fields is not tested; this list is based on the researcher's intuition. In a closer setting, the next section discusses the issue of impact of the concept on after-sales service.

### 8.4.1 IMPACT OF THE CONCEPT

Within the organization, the impact of the concept could very well extend beyond its original design. The concept is conceptualized for facilitating installations but this is by no means its only relevance. After-sales service in general is its playing field; here, Kefec could have high impact on four out of six activities (Figure 157).

| Risk reducing activities and the supporting process | Warranties / Service contracts | | | | | |
|---|---|---|---|---|---|---|
| Activities that are required to solve customers needs in after-sales service and their supporting processes | Installation | Training | Routine maintenance | Emergency repair | Parts supply | Software service |
| Product design | Physical product | | | | | |

**Figure 157 - Impact of embedded configuration on after-sales service**

Impact on repairs, and not least software service, can be huge. Restructuring knowledge with abstractions could change how software is used in embedded form, making it more tolerant of new versions and capable of transferring knowledge when

making repairs. When the Kefec is truly complete, it would be very interesting to make these thoughts into a proposition and test if they actually hold. The researchers' acknowledge that there are some limitations that have to be overcome before this is possible.

### 8.4.2 LIMITATIONS

In the beginning, we made some assumptions. Some of these were necessary for the work to proceed. It is worth wondering about what the limitations are and how they can eventually be overcome. The limitation could also be called scientific implications. Let us look at some points worthy of inspection.

- Strong focus on literature, the need to strengthen background. The search for literature has been largely based on intuition, and some important domains might be left out. Like everyone else, we are limited and rely on our experience to guide us. So, of course that colours our choices. A peer review on the suggestion here could open up new areas of inspiration and further this research.
- The suggestions here are not tested enough, both in regard to effectiveness and efficiency. More iteration is needed to perfect them. This is highly rooted in the conceptual nature of the work; without the concepts properly in place, it is hard to test.
- This research is grounded theory, based on a single case. Studying other cases that fulfil the same criterion for the environment should validate the generality created here.
- The work here focuses on meaning and assumes that communication is possible, both in regard to physical carrier and protocol. Physical and protocol layers have to be made and agreed upon before any practical prototype can be made.
- The consequence of this last point could be the cost of implementation, both in terms of hardware and organizational changes. The current hardware installed in the products could have performance problems and not enough processing power or memory storage to facilitate the embedded models. So a cost-benefit analysis of implementing the method should be made. But, as computer hardware still has an 18-month-doubling of capacity, the researchers doubt that this is going to be a prolonged problem.
- This work has been formulated as a fact-finding mission. It is not a theory (yet ☺).

In concluding this section on limitations, it is appropriate with some wonderings about theory. This work has not been about making theory. But, as the work has progressed, a light began to shine. There seems to be an underlying proposition in this work that is particularly in line with making theory. The work is about decision variables and how to reduce them. This allows us to speculate about whether there might be a theory here. Let us try to formulate the core of this thesis as a theory.

## 8.5 THE THEORY OF DECISION VARIABLES

This is about formulating our research as if its goal was to make new theory. The findings presented here are the result of our fact-finding mission and do not comprise a

complete theory. We have only placed the snippets in the appropriate places and structured it as well as we could. As we discussed in the opening chapter of this thesis (chapter 1.3 Research method), academics define a theory as being made up of four components: definitions, domains, relations and predictions. Let us place our findings in these components and see how it looks.

### 8.5.1 DEFINITIONS OF TERMS OR VARIABLES

The unit of analysis here is the system. A system is constructed of nearly independent sub-systems. The system has a boundary between its inner workings and the environment. For each sub-system, description of a product can be made on three abstraction levels. The abstraction levels are: *application,* where the need for the product is described; *function,* where functionality of the product is described, both in terms of effects (intended functions) and technical functions; and the last level is *artefact* where the product is described in terms of its parts. Each level has variables that control the final functionality. *Decision variable* is a variable that transcends the system boundary and requires an *external action*.

### 8.5.2 A DOMAIN WHERE THE THEORY APPLIES

This is about man-made things, artefacts or products. It is about artificial things that show the following characteristics. The artefact or system of artefacts is a multifunctional artefact, where variance or functionality is controlled with variables that are set by an external action. The variables can be both mechanical and software in nature.

### 8.5.3 A SET OF RELATIONSHIPS OF VARIABLES

An application is a group of functions, and functions are realized through one or several parts of an artefact. An item on application, function and artefact level can be a variable. The variables are linked internally on one level and between levels to create cascading effects, from application to artefact.

### 8.5.4 SPECIFIC PREDICTIONS (FACTUAL CLAIMS)

The number of Decision Variables needed in a system to control its functionality decreases drastically as they are placed higher in the abstractions, from artefact up to application.

## 8.6 CONCLUSIONS

This thesis presents a way to simplify the setup of complex product systems with the help of embedded configuration. To achieve this, the focus has to be on what subsystems need to communicate between themselves to structure the required internal knowledge and to form a communication protocol. The simplification of internal workings is due both to hardware- and application-induced configuration that takes

place both within the overall system and in each sub-system. By relating parameters in such a way, user inputs or decision variables should decrease drastically, and the overall usability of the installation increases. In our case, we have rationalized that this should be done with embedded configuration and the expected result is enhanced usability. The next step can be said to be two-fold: first, construct a system based on this philosophy and show that it actually leads to the expected results; and second, further develop the modelling tools and methods for supporting the making of embedded configuration systems or, in essence, a distributed artificial intelligence system.

The suggested method is highly rooted in system theory. It draws on the emergent properties expected from the system and tries to embed the knowledge needed to achieve them, into the system. In order to understand the system, we draw simplified functional streams, identify archetypes from the product assortment, and then map the two together into a system breakdown model. The system model points to how many encapsulation models should be made and the first decomposition in their tree centrepiece. The encapsulation model describes the archetype in three abstraction levels: application, function, and the physical artefact. All levels are connected through relational matrixes, both for internal and mapping relations. The models are stringent and thought out to be implemented in software. They should allow both import and export of product knowledge from a knowledge-based system.

The purpose of this work is to simplify the installation process of product systems that have been treated with extreme postponement, meaning that variance is defined with software variables when installing. These variables are defined as decision variables, and it is their reduction that is the overall goal.

### 8.6.1 OUTSET

As with all planning, the outset of this research was ambiguous: to construct a theoretically based suggestion for solving a very huge practical problem at the case company. At first, the plan was to use prototypes to evolve a solution. It became apparent quite early that making "random" prototypes would not create adequately drastic solutions. Meaning, that such prototypes would probably end up being monolithic IT systems that burden the organization. There was more value though in focusing on theoretically founded concepts that could really change some things. But this of course turned out to be much more time-consuming than expected. Thus, the testing never reached a whole system prototype, also because the case company was not willing to allocate resources for such work. The result is a modelling technique, hopefully well founded in the academia and tested on the knowledge level.

### 8.6.2 PROMISING FRAMEWORK

The researchers are actually very pleased with this work. It has been really pleasurable to find much similar thinking in the literature and to confirm that the intuitive thoughts that started all this were both relevant and promising. Especially promising is the connection to *immobots*; the rationale presented by (Williams & Pandurang Nayak 1996) is very close to that of the researchers!

## 8.7 NEXT STEPS – FUTURE WORK

While writing these last words, the researchers are smiling. Even though the process has been challenging and sometimes hard, it has been rewarding. Based on the suggestions made in this thesis, we can see several possible next steps that could further this work. Let us list them in bullet form with short explanations:

- *Relational testing*: Better connection should be created between the actual (current) list of relations and all those suggested in Mereology. The theoretical work should aid in creating even more general relations and help further the work of product modelling.
- *Effect Basis*: Suggest effect basis for the EM. Effect is an intended function of an artefact. A basis exists for technical functions (the Functional Basis) but none for the effects. To make the EM more attractive and easier to use, a complete effect basis that can be used to describe intended functionality of an artefact should be suggested.
- *Application construction*: Application construction is a core suggestion here. Its construction is fussy at the moment. How can the knowledge needed be collected? Or design intentions captured properly? Capturing customer needs is also necessary. But most of all, it is necessary to formulate how to model all these aspects into the EM and design a more formal process for the task.
- *Service basis*: Suggest service basis for the EM. Service is here in the form of module of an application. A single application is a collection of several services that then are made out of several effects. This should help in creating reuse in designs and facilitate redundancy management.
- *Colour palette*: Investigate colour palette usage and other analysing facilities of the models in more detail. The researchers believe a lot of potential exists here.
- *Communication model*: The communication model is not completed. We need to further it, prove the concept, test models, finish the process, and try to utilize SL and other elements from agent theory to put the finishing touches on the model.
- *Theory*: Create the theory of decision variables and test it.
- *Prototype*: Construction of an artefact prototype, maybe in form of immobots, with all the aspects described here, and test if it supports the rationale.

And this concludes the thesis, *Knowledge Engineering for Embedded Configuration*, thank you.

# TABLE OF FIGURES

# LITERATURE


1. Ackoff, R. L. 1971, "Towards a system of systems concepts", *Management Science*, vol. 17, no. 11, pp. 661-671.

2. Ackoff, R. L. 1973, "Science in the Systems Age: Beyond IE, OR, and MS*", *Operations Research*, vol. 21, no. 3, p. 661.

3. Ahmed, S., Blessing, L., & Wallace, K. "The Relationships between data, information and knowledge based on a preliminary study of engineering designers", in *ASME Design Theory and Methodology*, ASME, Las Vegas, Nevada, USA.

4. Ahmed, S., Kim, S., & Wallace, K. M. 2007, "A Methodology for Creating Ontologies for Engineering Design", *Transactions of the ASME - S - Computing and Information Science in Engineering* no. 2, p. 132.

5. Akao, Y. 1990, *Quality function deployment Integrating customer requirements into product design* Productivity Press, Cambridge, Massachuesett.

6. Akkermans, H., van Harmelen, F., Schreiber, G., & Wielinga, B. 1993, "A formalization of knowledge-level models for knowledge acquisistion", *International Journal of Intelligent Systems*, vol. 8, no. 2, pp. 169-208.

7. Aler, R., Borrajo, D., Camacho, D., & Sierra-Alonso, A. 2002, "A knowledge-based approach for business process reengineering, SHAMASH", *Knowledge-Based Systems*, vol. 15, no. 8, pp. 473-483.

8. Allmendinger, G. & Lombreglia, R. 2005, "Four strategies for the age of smart services", *Harvard Business Review*, vol. 83, no. 10, pp. 131-145.

9. Alonso-Rasgado, T., Abdelkafi, N., Thompson, G., & Elfstrom, B. O. 2004, "The design of functional (total care) products", *Journal of Engineering Design*, vol. 15, no. 6, pp. 515-540.

10. Andreasen, M. M. 1980, *Syntesemetoder på systemgrundlag - Bidrag til en konstruktions teori [in Danish],* Department of Machine Design, Lund Institute of Technology.

11. Andreasen, M. M. 1991, "Design methodology", *Journal of Engineering Design*, vol. 2, no. 4, pp. 321-335.

12. Andreasen, M. M.  15-8-2007.
Ref Type: Personal Communication

13. Artale, A., Franconi, E., Guarino, N., & Pazzi, L. 1996, "Part-whole relations in object-centered systems: An overview", *Data and Knowledge Engineering*, vol. 20, no. 3, pp. 347-383.

14. Arthur, J. D. & Gröner, M. K. 2005, "An operational model for structuring the requirements generation process", *Requirements Engineering*, vol. 10, no. 1, pp. 45-62.

15. Ashby, W. R. 1956, *An introduction to Cybernetics* Chapman & Hall ltd, London, UK.

16. Ashby, W. R. 1973, "Some peculiarities of complex systems", *Cybernetic Medicne*, vol. 9, no. 2, pp. 1-6.

17. Baglietto, P., Maresca, M., Parodi, A., & Zingirian, N. 2005, "Stepwise deployment methodology of a service oriented architecture for business communities", *Information and Software Technology*, vol. 47, no. 6, pp. 427-436.

18. Bartolomei, J. E. 2007, *Qualitative Knowledge Construction for Engineering Systems: Extending the Design Structure Matrix Methodology in Scope and Procedure,* PhD Thesis, Massachusetts Institute of Technology, Engineering Systems Division.

19. Bertalanffy, L. V. 1969, *General system theory. Foundations, development, applications*, Revised repr.in 1973 ed. edn, Braziller, New York.

20. Bertalanffy, L. V. 1972, "The History and Status of General Systems Theory", *Academy of Management Journal*, vol. 15, no. 4, pp. 407-426.

21. Bi, Z. M. & Zhang, W. J. 2001, "Modularity Technology in Manufacturing: Taxonomy and Issues", *International Journal of Advanced Manufacturing Technology*, vol. 18, no. 5, pp. 381-390.

22. Biggs, J. B. & Collis, K. F. 1982, *Evaluating the Quality of Learning – the SOLO Taxonomy* Academic Press, New York.

23. Bittner, T. & Donnelly, M. 2005, "Spatial Relations Between Classes of Individuals", *Lecture Notes in Computer Science*, vol. 3693, pp. 182-199.

24. Blakemore & Frith 2005, *The learning brain* Blackwell Publishing.

25. Blessing, L. 2002, "What is this thing called Design Research?", in *Int'l CIRP Design Seminar*.

26. Bloom, B. S. 1956, *Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain* Longmans Group, New York and Toronto.

27. Borst, P., Akkermans, H., & Top, J. 1997, "Engineering ontologies", *International Journal of Human-Computer Studies*, vol. 46, no. 2-3, pp. 365-406.

28. Boulding, K. E. 1956, "General Systems Theory-The Skeleton of Science", *Management Science*, vol. 2, no. 3, pp. 197-208.

29. Browning, T. R. 2001, "Applying the design structure matrix to system decomposition and integration problems: a review and new directions", *IEEE Transactions on Engineering Management*, vol. 48, no. 3, pp. 292-306.

30. Buede, D. 2000, *The Engineering Design of Systems - Models and methods* Wiley, New York.

31. Buhne, S., Lauenroth, K., & Pohl, K. 2005, "Modelling Requirements Variability across Product Lines", *Requirements Engineering, 2005.Proceedings.13th IEEE International Conference on* pp. 41-52.

32. Buur, J. & Andreasen, M. M. 1989, "Design models in mechatronic product development", *Design Studies*, vol. 10, no. 3, pp. 155-162.

33. Casati, R. & Varzi, A. C. 1999, *Parts and Places: The Structures of Spatial Representation* The MIT Press, Cambridge, Massachusetts.

34. Chakrabarti, A., Sarkar, P., Leelavathamma, B., & Nataraju, B. S. 2005, "A functional representation for aiding biomimetic and artificial inspiration of new ideas", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 19, no. 2, pp. 113-132.

35. Chambers, J. C. & Mullick, S. K. 1971, "How to choose the right forecasting technique", *Harvard Business Review*, vol. 49, no. 4, pp. 45-70.

36. Chandrasekaran, B. 1990, "Design problem solving: a task analysis", *AI Magazine*, vol. 11, no. 4, pp. 59-71.

37. Chandrasekaran, B., Goel, A. K., & Iwasaki, Y. 1993, "Functional representation as design rationale", *Computer*, vol. 26, no. 1, pp. 48-56.

38. Checkland, P. 1984, *Systems thinking, systems practice* Wiley, New York.

39. Chen, C. H. & Rao, Z. 2005, "A Matrix Representation and Mapping Approach to Knowledge Acquisition for Product Design", *Lecture Notes in Computer Science*, vol. 3681, pp. 311-317.

40. Churchman, C. W. 1968, *the System Approach* Dell Publishing Co. Inc., New York.

41. Clarke, B. R. 1989, "Knowledge-based configuration of industrial automation systems", *International Journal of Computer Integrated Manufacturing*, vol. 2, no. 6, pp. 345-355.

42. Cohen, M. A., Agrawal, N., & Agrawal, V. 2006, "Winning the aftermarket", *Harvard Business Review*, vol. 84, no. 5, pp. 129-138.

43. Cohen, M. A. & Lee, H. L. 1990, "Out of Touch with Customer Needs? Spare Parts and After Sales Service", *Sloan Management Review*, vol. 31, no. 2, pp. 55-66.

44. Cohn, A. G. & Varzi, A. C. 2003, "Mereotopological Connection", *Journal of Philosophical Logic*, vol. 32, no. 4, pp. 357-390.

45. Coughlan, P. & Coghlan, D. 2002, "Action research for operations management", *International Journal of Operations and Production Management*, vol. 22, no. 2, pp. 220-240.

46. Coulouris, G., Dollimore, J., & Kindberg, T. 2005, *Distributed systems: Concepts and design*, 4 edn, Addison-Wesley, Harlow, England.

47. Da Silveira, G., Borenstein, D., & Fogliatto, F. S. F. 2001, "Mass customization: Literature review and research directions", *International Journal of Production Economics*, vol. 72, no. 1, pp. 1-13.

48. Danilovic, M. & Browning, T. R. 2007, "Managing complex product development projects with design structure matrices and domain mapping matrices", *International Journal of Project Management*, vol. 25, no. 3, pp. 300-314.

49. Davenport, T. H. & Prusak, L. 1998, *Working Knowledge How Organizations Manage What They Know.* Harvard Business School Press, Boston, Mass.

50. Davenport, T. H. & Short, J. E. 1990, "The new industrial engineering: information technology and business process redesign", *Sloan Management Review*, vol. 31, no. 4, pp. 11-27.

51. de Kleer, J. & Brown, J. S. 1984, "A qualitative physics based on confluences", *Artificial Intelligence*, vol. 24, no. 1-3, pp. 7-83.

52. Dervin, B. 1998, "Sense-making theory and practice: an overview of user interests in knowledge seeking and use", *Journal of Knowledge Management*, vol. 2, no. 2, pp. 36-46.

53. Duray, R., Ward, P. T., Milligan, G. W., & Berry, W. L. 2000, "Approaches to mass customization: configurations and empirical validation", *Journal of Operations Management*, vol. 18, no. 6, pp. 605-625.

54. Dutoit, A. H. & Paech, B. 2002, "Rationale-Based Use Case Specification", *Requirements Engineering*, vol. 7, no. 1, pp. 3-19.

55. Erens, F. & McKay, A. 1994, "Product modelling using multiple levels of abstraction instances as types", *Computers in Industry*, vol. 24, no. 1, pp. 17-28.

56. Erixon, G. 1998, *Modular function deployment,* PhD Thesis, Institutionen för produktionssystem, Royal Institute of Technology (KTH).

57. Erixon, G., von Yxkull, A., & Arnstrom, A. 1996, "Modularity - the basis for product and factory reengineering", *CIRP Annals - Manufacturing Technology*, vol. 45, no. 1, pp. 1-4.

58. Ethiraj, S. K. & Levinthal, D. 2004, "Modularity and Innovation in Complex Systems", *Management Science*, vol. 50, no. 2, pp. 159-173.

59. Evermann, J. & Wand, Y. 2005, "Ontology based object-oriented domain modelling: fundamental concepts", *Requirements Engineering*, vol. 10, no. 2, pp. 146-160.

60. Felfernig, A., Friedrich, G. E., & Jannach, D. 2000, "UML as Domain Specific Language for the Construction of Knowledge-based Configuration Systems", *International Journal of Software Engineering & Knowledge Engineering*, vol. 10, no. 4, p. 449.

61. FIPA 2001, *Ontology service specification* XC00086D.

62. FIPA 2002a, *Abstract architecture specification* SC00001L.

63. FIPA 2002b, *ACL message structure specification* SC00061G.

64. FIPA 2002c, *Communicative Act Library specification* SC00037J.

65. FIPA 2002d, *Device Ontology Specification* SI00091E.

66. FIPA 2002e, *SL Content Language Specification* SC00008I.

67. Fixson, S. K. 2003, *The Multiple Faces of Modularity - A literature analysis of a product concept for assembled hardware products*, University of Michigan.

68. Forza, C. & Salvador, F. 2007, *Product Information Management for Mass Customization: Connecting customer, front-office and back-office for fast and efficient customization* Palgrave.

69. Forza, C., Salvador, F., & Trentin, A. "Form postponement from a decision-making perspective", in *3rd Interdisciplinary World Congress on Mass Customization and Personalization, Hong Kong, 18-21 september 2005*.

70. Genesereth, M. R. & Fikes, R. E. 1992, *Knowledge Interchange Format, version 3*, Computer Science Department, Stanford University, Stanford, California US.

71. Geoffrion, A. M. 1989, "Computer-based modeling environments", *European Journal of Operational Research*, vol. 41, no. 1, pp. 33-43.

72. Gero, J. S. 1990, "Design prototypes. A knowledge representation schema for design", *AI Magazine*, vol. 11, no. 4, pp. 26-36.

73. Gershenson, J. K., Prasad, G. J., & Zhang, Y. 2003, "Product modularity: definitions and benefits", *Journal of Engineering Design*, vol. 14, no. 3, pp. 295-313.

74. Gerstl, P. & Pribbenow, S. 1996, "A conceptual theory of part-whole relations and its applications", *Data and Knowledge Engineering*, vol. 20, no. 3, pp. 305-322.

75. Gilmore, J. H. & Pine, B. J. I. 1997, "The four faces of mass customization", *Harvard Business Review*, vol. 75, no. 1, pp. 91-101.

76. Glasersfeld, E. v. 1991, "An Exposition of Constructivism: Why some like it radical," in *Facets of system science*, G. J. Klir, ed., Plenum Press, New York, pp. 229-238.

77. Glasersfeld, E. v. 2001, "The Radical Constructivist View of Science", *Foundations of Science*, vol. 6, no. 1-3, pp. 31-43.

78. Gomez-Perez, A., Fernandez-Lopez, M., & Corcho, O. 2004, *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web* Springer-Verlag, London, UK.

79. Gorschek, T. & Wohlin, C. 2006, "Requirements Abstraction Model", *Requirements Engineering*, vol. 11, no. 1, pp. 79-101.

80. Gruber, T. R. 1993, "A translation approach to portable ontology specifications", *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220.

81. Gruber, T. R. 1995, "Toward principles for the design of ontologies used for knowledge sharing?", *International Journal of Human-Computer Studies*, vol. 43, no. 5-6, pp. 907-928.

82. Gu, P. & Sosale, S. 1999, "Product modularization for life cycle engineering", *Robotics and Computer-Integrated Manufacturing*, vol. 15, no. 5, pp. 387-401.

83. Guarino, N. 1997, "Understanding, building and using ontologies", *International Journal of Human-Computer Studies*, vol. 46, no. 2-3, pp. 293-310.

84. Gupta, S. & Krishnan, V. 1998, "Product family-based assembly sequence design methodology", *IIE Transactions*, vol. 30, no. 10, pp. 933-945.

85. Hammer, M. 1990, "Reengineering Work: Don't Automate, Obliterate", *Harvard Business Review*, vol. 68, no. 4, pp. 104-112.

86. Hammer, M. 2004, "Deep change - how operational innovation can transform your company", *Engineering Management Review, IEEE*, vol. 32, no. 3, p. 42.

87. Hansen, C. T. & Rutahuhta, A. 2001, "Issues on the development and application of computer tools to support product structuring and configuring", *International Journal of Technology Management*, vol. 21, no. 3-4, p. 240.

88. Harlou, U. 2006, *Developing product families based on architectures: Contribution to a theory of product families,* PhD Thesis, Technical University of Denmark (DTU).

89. Hartley, R. T. & Barnden, J. A. 1997, "Semantic networks: visualizations of knowledge", *Trends in Cognitive Sciences*, vol. 1, no. 5, pp. 169-175.

90. Haug, A. 2007, *Representation of Industrial Knowledge - as a Basis for Developing and Maintaining Product Configurators,* PhD Thesis, DTU Management, Technical University of Denmark.

91. He, D., Kusiak, A., & Tseng, T. L. 1998, "Delayed product differentiation: a design and manufacturing perspective", *Computer-Aided Design*, vol. 30, no. 2, pp. 105-113.

92. Hegge, H. M. H. & Wortmann, J. C. 1991, "Generic bill-of-material: a new product model", *International Journal of Production Economics*, vol. 23, no. 1-3, pp. 117-128.

93. Heinrich, M. & Jungst, E. W. "A resource-based paradigm for the configuring of technical systems from modular components", IEEE Comput. Soc. Press, pp. 257-264.

94. Helo, P. T. 2006, "Product configuration analysis with design structure matrix", *Industrial Management and Data Systems*, vol. 106, no. 7, pp. 997-1011.

95. Hicks, R. C. 2003, "Knowledge base management systems-tools for creating verified intelligent systems", *Knowledge-Based Systems*, vol. 16, no. 3, pp. 165-171.

96. Hippel, E. v. 1990, "Task Partitioning: An Innovation Process Variable", *Research Policy*, vol. 19, no. 5, pp. 407-418.

97. Hirtz, J., Stone, R. B., McAdams, D., Szykman, S., & Wood, K. 2002, "A functional basis for engineering design: Reconciling and evolving previous efforts", *Research in Engineering Design*, vol. 13, no. 2, pp. 65-82.

98. Hofer, A. P. & Halman, J. I. M. 2004, "Complex products and systems: Potential from using layout platforms", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 18, no. 1, pp. 55-69.

99. Hofstede, G. 2001, *Culture's Consequence: Comparing Values, Behaviours, Institutions and Organizations Across Nations*, 2 edn, SAGE Publications Inc, USA.

100. Hopgood, A. A. 2000, *Intelligent systems for engineers and scientists*, 2. edn, CRC Press, Boca Raton, FL.

101. Huang, C. C. & Kusiak, A. 1998, "Modularity in design of products and systems", *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, vol. 28, no. 1, pp. 66-77.

102. Hubka, V. & Eder, W. E. 1987, "A scientific approach to engineering design", *Design Studies*, vol. 8, no. 3, pp. 123-137.

103. Hubka, V. & Eder, W. E. 1996, *Design Science* Springer, London, Great Britain.

104. Hull, E., Jackson, K., & Dick, J. 2005, *Requirements Engineering*, 2 edn, Springer, London.

105. Hvam, L. 1999, "A procedure for building product models", *Robotics and Computer-Integrated Manufacturing*, vol. 15, no. 1, pp. 77-87.

106. Hvam, L. 2001, "A procedure for the application of product modelling", *International Journal of Production Research*, vol. 39, no. 5, pp. 873-885.

107. Hvam, L., Mortensen, N. H., & Riis, J. 2007, *Product Customization*.

108. IDEF 1993, *Integration Definition for Function Modeling (IDEF0)*, National Institute of Standards and Technology, USA.

109. IDEF 1994, *IDEF5 Method Report*, National Institute of Standards and Technology, USA, F33615-C-90-0012.

110. Iris, M., Lutowitz, B., & Evens, M. 1988, "Problems of the part-whole relation," in *Relational models of the lexicon*, Cambridge University Press, Cambridge, pp. 261-288.

111. ISO 2002, *Systems Engineering - System Life Cycle Processes*, ISO copyright office, Switzerland, ISO/IEC 15288.

112. ISO 2003, *Systems engineering - A guide for the application of ISO/IEC 15288 (System life cycle processes)*, ISO copyright office, Switzerland, ISO/IEC TR 19760.

113. Ives, B. & Vitale, M. R. 1988, "After the sale: leveraging maintenance with information technology", *Management Information Systems Quarterly*, vol. 12, no. 1, pp. 7-21.

114. Iwasaki, Y., Fikes, R., Vescovi, M., & Chandrasekaran, B. "How things are intended to work: capturing functional knowledge in device design", In *Int. Joint Conferences on Artificial Intelligence*, pp. 1516-1522.

115. Jensen, T. 1999, *Functional Modelling in a Design Support System,* PhD, Department of Control and Engineering Design.

116. Jiao, J. & Tseng, M. M. 2000, "Fundamentals of product family architecture", *Integrated Manufacturing Systems*, vol. 11, no. 7, pp. 469-483.

117. Johannesson, H. & Claesson, A. 2005, "Systematic product platform design: a combined function-means and parametric modeling approach", *Journal of Engineering Design*, vol. 16, no. 1, pp. 25-43.

118. Jørgensen, K. A. 1992, *Videnskabelige arbejdsparadigmer (in Danish)* Institut for Produktion, Aalborg Universitet, Danmark.

119. Juengst, W. E. & Heinrich, M. 1998, "Using resource balancing to configure modular systems", *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 4, pp. 50-58.

120. Kavakli, E. 2002, "Goal-Oriented Requirements Engineering: A Unifying Framework", *Requirements Engineering*, vol. 6, no. 4, pp. 237-251.

121. Kendal, S. & Creen, M. 2007, *An introduction to Knowledge Engineering* Springer-Verlag, London.

122. Kettinger, W. J., Teng, J. T. C., & Guha, S. 1997, "Business Process change: A study of methodologies, techniques, and tools", *MIS Quarterly: Management Information Systems*, vol. 21, no. 1, pp. 55-80.

123. Kimura, F., Kato, S., Tomoyuki, H., & Masuda, T. 2001, "Product modularization for parts reuse in inverse manufacturing", *CIRP Annals - Manufacturing Technology*, vol. 50, no. 1, pp. 89-92.

124. Kirschman, C. F. & Fadel, G. M. 1998, "Classifying functions for mechanical design", *Journal of Mechanical Design, Transactions of the ASME*, vol. 120, no. 3, pp. 475-482.

125. Kitamura, Y. & Mizoguchi, R. 2004a, "Ontology-based systematization of functional knowledge", *Journal of Engineering Design*, vol. 15, no. 4, pp. 327-351.

126. Kitamura, Y. & Mizoguchi, R. 2004b, "Ontology-based functional-knowledge modeling methodology and its deployment", *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, vol. 3257, pp. 99-115.

127. Klir, G. J. 2001, *Facets of systems science*, 2 edn, Kluwer Academic / Plenum Publishers, New York.

128. Kotler, P. 1989, "From Mass Marketing To Mass Customization", *Planning Review*, vol. 17, pp. 10-14.

129. Krause, F. L., Kimura, F., Kjellberg, T., Lu, S. C. Y., van der Wolf, A. C. H., Ating, L., ElMaraghy, H. A., Eversheim, W., Iwata, K., Suh, N. P., Tipnis, V. A., & Weck, M. 1993, "Product modelling", *CIRP Annals*, vol. 42, no. 2, pp. 695-706.

130. Kuhn, T. S. 1996, *The Structure of the Scientific Revolution*, 3 edn, The University of Chicago Press, Chicago and London.

131. Kusiak, A. & Larson, N. 1995, "Decomposition and representation methods in mechanical design", *Journal of Mechanical Design, Transactions of the ASME*, vol. 117B, pp. 17-24.

132. Kusiak, A. & Salustri, F. A. 2007, "Computational Intelligence in Product Design Engineering: Review and Trends", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 5, pp. 766-778.

133. Lampel, J. & Mintzberg, H. 1996, "Customizing Customization", *Sloan Management Review*, vol. 38, no. 1, pp. 21-30.

134. Langlois, R. N. & Robertson, P. 1992, "Networks and Innovation in a Modular System: Lessons from the Microcomputer and Stereo Component Industries", *Research Policy*, vol. 21, no. 4, pp. 297-313.

135. Leite, J. C. S. d. P., Doorn, J. H., Hadad, G. D. S., & Kaplan, G. N. 2005, "Scenario inspections", *Requirements Engineering*, vol. 10, no. 1, pp. 1-21.

136. Levesque, H. J. 1984, "Foundations of a functional approach to knowledge representation", *Artificial Intelligence*, vol. 23, no. 2, pp. 155-212.

137. Lewis, D. W. 2001, *Fundamentals of Embedded software: where C and Assembly meet* Prentice-Hall.

138. Li, Z. & Ramani, K. 2007, "Ontology-based design information extraction and retrieval", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 21, no. 2, pp. 137-154.

139. Li, Z., Raskin, V., & Ramani, K. "Developing ontologies for engineering information retrival", in *ASME Design Engineering Technical Conference*, Las Vegas, Nevada, USA.

140. Lind, M. 1990, *Representing goals and functions of complex systems*, Institute of Automatic Control Systems, Technical University of Denmark, Copenhagen, Denmark, Technical Report 90-D-381.

141. Lind, M. 1994, "Modeling goals and functions of complex industrial plants", *Applied Artificial Intelligence*, vol. 8, no. 2, pp. 259-283.

142. Lossack, R.-S. "Foundations for a Universal Design Theory - A design process model", in *INSA*, Lyon, France.

143. Magro, D. & Torasso, P. 2003, "Decomposition strategies for configuration problems", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 17, no. 1, pp. 51-73.

144. Malmqvist, J. 1997, "Improved function-means trees by inclusion of design history information", *Journal of Engineering Design*, vol. 8, no. 2, pp. 107-117.

145. Malmqvist, J. "A classification of matrix-based methods for product modeling", in *Design 2002*.

146. Mannisto, T., Peltonen, H., Soininen, T., & Sulonen, R. 2001, "Multiple abstraction levels in modelling product structures", *Data and Knowledge Engineering*, vol. 36, no. 1, pp. 55-78.

147. Martin, M. & Ishii, K. 2002, "Design for variety: developing standardized and modularized product platform architectures", *Research in Engineering Design*, vol. 13, no. 4, pp. 213-235.

148. Maslow, A. 1943, "A theory of Human Motivation", *Psychological Review*, vol. 50, no. 4, pp. 370-396.

149. Mead, G. H. 1938, *The Philosophy of the Act* University of Chicago Press, Chicago.

150. Meredith, J. R. 1989, "Alternative Research Paradigms in Operations", *Journal of Operations Management*, vol. 8, no. 4, p. 297.

151. Meredith, J. R. 1998, "Building operations management theory through case and field research", *Journal of Operations Management*, vol. 16, no. 4, pp. 441-454.

152. Mikkola, J. H. & Skjott-Larsen, T. 2004, "Supply-chain integration: Implications for mass customization, modularization and postponement strategies", *Production Planning & Control*, vol. 15, no. 4, pp. 352-361.

153. Mili, F., Shen, W., Martinez, I., Noel, P., Ram, M., & Zouras, E. 2001, "Knowledge modeling for design decisions", *Artificial Intelligence in Engineering*, vol. 15, no. 2, pp. 153-164.

154. Miller, G. A. 1994, "The magical number seven, plus or minus two: Some limits on our capacity for processing information", *Psychological Review*, vol. 101, no. 2, pp. 343-352.

155. Miller, T. D. 2001, *Modular Engineering: An approach to structuring business with coherent, modular architectures of artifacts, activities and knowledge,* PhD Thesis, Department of Mechanical Engineering, Technical University of Denmark (DTU).

156. Mittal, S. & Frayman, F. "Towards a generic model of configuration tasks", Int. Joint Conferences on Artificial Intelligence, pp. 1395-1401.

157. Mizoguchi, R., Tijerino, Y., & Ikeda, M. 1995a, "Task Analysis Interview Based on Task Ontology", *Expert Systems with Applications*, vol. 9, no. 1, pp. 15-25.

158. Mizoguchi, R., Vanwelkenhuysen, J., & Ikeda, M. "Task ontology for reuse of problem solving knowledge", IOS Press, pp. 46-59.

159. Modarres, M. "Functional modeling of Physical systems using the Goal tree-success tree framework", in *Proceedings of the First International Workshop on Functional Modeling of Complex Technical Systems*, Ispra, Italy.

160. Modarres, M. & Cheon, S. W. 1999, "Function-centered modeling of engineering systems using the goal tree-success tree technique and functional primitives", *Reliability Engineering and System Safety*, vol. 64, no. 2, pp. 181-200.

161. MOKA 2000, *Methodology and tools Oriented to Knowledge-based engineering Applications (MOKA) - MOKA User guide* D1.3-Annex B (MOKA/COV/TK1.5/DL/D1.3B/1/CC/SUB).

162. Mortensen, N. H. 1999, *Design modelling in a designer's workbench, contribution to a Design Language,* Technical University of Denmark.

163. Mortensen, N. H. & Hansen, C. T. 1999, "Structuring as a basis for Product Modelling," in *Critical Entusiasm, Contributions to Design Science*, Tapir, Trondheim, Norway, pp. 111-128.

164. Mueller & Schappert 1999, *The Knowledge Factory - A Generic Knowledge Management Architecture*.

165. Newcomb, P. J., Bras, B., & Rosen, D. W. 1998, "Implications of modularity on product Design for the Life Cycle", *Journal of Mechanical Design, Transactions of the ASME*, vol. 120, no. 3, pp. 483-491.

166. Newell, A. 1982, "The knowledge level", *Artificial Intelligence*, vol. 18, no. 1, pp. 87-127.

167. Newell, A. 1990, *Unified Theories of Cognition* Harvard University Press, Cambridge, Massachusetts.

168. Nonaka, I. 1991, "The Knowledge-Creating Company", *Harvard Business Review*, vol. 69, no. 6, pp. 96-104.

169. Nonaka, I. 1994, "A Dynamic Theory of Organizational Knowledge Creation", *Organization Science*, vol. 5, no. 1, pp. 14-37.

170. Noy, N. F. & McGuinness, D. L. 2001, *Ontology Development 101: A Guide to Creating Your First Ontology*, Stanford University, CA, USA, Technical Report, SMI-2001-0880.

171. O'Donnell, F. J., MacCallum, K. J., Hogg, T. D., & Yu, B. 1996, "Product structuring in a small manufacturing enterprise", *Computers in Industry*, vol. 31, no. 3, pp. 281-292.

172. Olhager, J. 2003, "Strategic positioning of the order penetration point", *International Journal of Production Economics*, vol. 85, no. 3, pp. 319-329.

173. OMG 2007, *SysML - Systems Modeling Language*.

174. Orsvarn, K. 1998, "Some principles for libraries of task decomposition methods", *International Journal of Human Computer Studies*, vol. 49, no. 4, p. 417.

175. Pagh, J. D. & Cooper, M. C. 1998, "Supply Chain Postponement and Speculation Strategies: How to Choose the Right Strategy", *Journal of Business Logistics - Council of Logistics Management*, vol. 19, no. 2, pp. 13-34.

176. Pahl, G., Beitz, W., Feldhusen, J., & Grote, K. H. 2007, *Engineering Design – a systematic approach*, 3 edn, Springer-Verlag.

177. Patnaik, D. & Becker, R. 1999, "Need finding: The Why and How of Uncovering People's Needs", *Design Management Journal*, vol. 10, no. 2, p. 37.

178. Piller, F. T. 2005, "Mass Customization: Reflections on the State of the Concept", *International Journal of Flexible Manufacturing Systems*, vol. 16, no. 4, pp. 313-334.

179. Pimmler, T. U. & Eppinger, S. D. 1994, "Integration analysis of product decompositions", *6th International Conference on Design Theory and Methodology*, vol. 68, pp. 343-351.

180. Pine II, B. J. 1993, *Mass customization: The new frontier in business competition* Harvard Business School Press, Boston,Massachusett.

181. Polanyi, M. 1962, "Tacit knowing. Its bearing on some problems of philosophy", *Reviews of Modern Physics*, vol. 34, no. 4.

182. Popper, K. 2002, *The Logic of Scientific Discovery*, 2 edn, Routledge Classic, London and New York.

183. Potts, G. W. 1988, "Exploit Your Product's Service Life Cycle", *Harvard Business Review*, vol. 66, no. 5, pp. 32-35.

184. Power, Y. & Bahri, P. A. 2005, "Integration techniques in intelligent operational management: a review", *Knowledge-Based Systems*, vol. 18, no. 2-3, pp. 89-97.

185. Qian, L. & Gero, J. S. 1996, "Function-behavior-structure paths and their role in analogy-based design", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 10, no. 4, pp. 289-312.

186. Rauterberg, G. W. M. The new economy: e-commerce, intellectual property rights, trust and other dangerous things. 2001. slides from ICT congress in Netherlands. Ref Type: Slide

187. Rosenman, M. A. & Gero, J. S. 1996, "Modelling multiple views of design objects in a collaborative CAD environment", *Computer-Aided Design*, vol. 28, no. 3, pp. 193-205.

188. Rumbaugh, J., Jacobson, I., & Booch, G. 2005, *The unified modeling language reference manual*, 2. edn, Addison-Wesley, Boston, Mass.

189. Russell, S. J. & Norvig, P. 2003, *Artificial Intelligence: A Modern Approach*, 2 edn, Pearson Education, Prentice Hall, Upper Saddle River, New Jersey, USA.

190. Sage, A. P. & Armstrong Jr, J. E. 2000, *Introduction to Systems Engineering* Wiley, New York.

191. Salles, J., Baranauskas, M. C., & Bigonha, R. S. 2001, "Towards a communication model applied to the interface design process", *Knowledge-Based Systems*, vol. 14, no. 8, pp. 455-459.

192. Salustri, F. A. & Venter, R. D. 1992, "Axiomatic theory of engineering design information", *Engineering with Computers (New York)*, vol. 8, no. 4, pp. 197-211.

193. Salustri, F. A. 2002, "Mereotopology for product modelling. A new framework for product modelling based on logic", *Journal of Design Technology*, vol. 2, no. 1.

194. Salvador, F. 2007, "Toward a Product System Modularity Construct: Literature Review and Reconceptualization", *IEEE Transactions on Engineering Management*, vol. 54, no. 2, pp. 219-240.

195. Salvador, F., Forza, C., & Rungtusanatham, M. 2002, "How to mass customize: Product architectures, sourcing configurations", *Business Horizons*, vol. 45, no. 4, pp. 61-69.

196. Sanchez, R. & Collins, R. P. 2001, "Competing-and Learning-in Modular Markets", *Long Range Planning*, vol. 34, no. 6, pp. 645-667.

197. Schilling, M. A. 2000, "Toward a General Modular Systems Theory and Its Application to Interfirm Product Modularity", *Academy of Management Review*, vol. 25, no. 2, pp. 312-334.

198. Schreiber, G., Akkermans, H., Anjewierden, A., Hoog, R. d., Shadbolt, N., van de Velde, W., & Wielinga, B. 2000, *Knowledge Engineering and Management: The CommonKADS Methodology* The MIT Press, Cambrigde, Massachusetts, USA.

199. Scott, R. W. 2003, *Organizations: Rational, Natural and Open Systems*, 5 edn, Prentice Hall, Englewood Cliffs, N.J.

200. Searle, J. 1969, *Speech Acts* Cambridge University Press.

201. Shannon, C. E. & Weaver, W. 1949, *Mathematical theory of communication* University of Illinois Press, USA.

202. Shupe, J. A., Mistree, F., & Sobieszanski-Sobieski, J. 1987, "Compromise: An effective approach for the hierarchical design of structural systems", *Computers and Structures*, vol. 26, no. 6, pp. 1027-1037.

203. Sim, S. K. & Duffy, A. H. B. 2003, "Towards an ontology of generic engineering design activities", *Research in Engineering Design*, vol. 14, no. 4, pp. 200-223.

204. Simon, H. A. 1945, *Administrative Behaviour*, 4 edn, Free Press, New York.

205. Simon, H. A. 1978, "Rationality as process and as product of thought", *The American Economic Review*, vol. 68, no. 2, p. 1.

206. Simon, H. A. 1991, "Artificial intelligence: where has it been, and where is it going?", *IEEE Transactions on Knowledge and Data Engineering*, vol. 3, no. 2, pp. 128-136.

207. Simon, H. A. 1995, "Artificial intelligence: an empirical science", *Artificial Intelligence*, vol. 77, no. 1, pp. 95-127.

208. Simon, H. A. 1996, *The Science of the Artificial*, 3 edn, The MIT press, Cambridge, Massachusetts.

209. Simon, H. A. 2002, "Organizing and coordinating talk and silence in organizations", *Industrial and Corporate Change*, vol. 11, no. 3, pp. 611-618.

210. Simons, P. 1987, *Parts: A study in ontology* Clarendon Press, Oxford, England.

211. Skyttner, L. 2001, *General systems theory* World Scientific Publishing Co. Pte. Ltd, Singapore.

212. Smith, B. 1996, "Mereotopology: A theory of parts and boundaries", *Data and Knowledge Engineering*, vol. 20, no. 3, pp. 287-303.

213. Smut, J. C. 1926, *Holism and Evolution*, Reprint 1999 edn, Kessinger Publishing, Whitefish, MT.

214. Snavely, G. L. & Papalambros, P. Y. 1993, "Abstraction as a configuration design methodology", *Advances in Design Automation*, vol. 65 pt 1, pp. 297-305.

215. Soerensen, M. U. 1999, "Application of functional modelling in the design of industrial control systems", *Reliability Engineering and System Safety*, vol. 64, no. 2, pp. 301-315.

216. Soininen, T., Tiihonen, J., Mannisto, T., & Sulonen, R. 1998, "Towards a general ontology of configuration", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 12, no. 4, pp. 357-372.

217. Sommerville, I. 2007, *Software Engineering*, 8 edn, Addison-Wesley, Harlow, England.

218. Starr, M. K. 1965, "Modular Production--A New Concept", *Harvard Business Review*, vol. 43, no. 6, pp. 131-142.

219. Stenmark, D. "Information vs. Knowledge: The Role of intranets in Knowledge Management", in *Hawaii International Conference on System Science*, IEEE, Proceedings of the 35th Hawaii International Conference on System Science.

220. Stevens, R., Brook, P., Jackson, K., & Arnold, S. 1998, *Systems Engineering: Coping with complexity* Prentice Hall Europe, London.

221. Steward, D. V. 1981, "The design structure system: a method for managing the design of complex systems", *IEEE Transactions on Engineering Management*, vol. EM-28, no. 3, pp. 71-74.

222. Stone, R. B. 1997, *Towards a theory of modular design,* PhD, University of Texas, Austin.

223. Stone, R. B. & Wood, K. L. 2000, "Development of a functional basis for design", *Journal of Mechanical Design*, vol. 122, no. 4, pp. 359-370.

224. Storga, M., Andreasen, M. M., & Marjanovic, D. "Towards a formal design model based on a genetic design model system", in *International Confereance on Engineering Design - ICED - 15 - 18 august*, Melbourne, Australia.

225. Strauss, A. & Corbin, J. 1990, *Basics of qualitative research. Grounded theory procedures and techniques* Sage Publications, Newbury Park, Cal.

226. Studer, R., Benjamins, V. R., & Fensel, D. 1998, "Knowledge Engineering: Principles and methods", *Data and Knowledge Engineering*, vol. 25, no. 1-2, pp. 161-197.

227. Stumptner, M. 1997, "An overview of knowledge-based configuration", *AI Communications*, vol. 10, no. 2, pp. 111-125.

228. Sugumaran, V. & Storey, V. C. 2002, "Ontologies for conceptual modeling: their creation, use, and management", *Data and Knowledge Engineering*, vol. 42, no. 3, pp. 251-271.

229. Suh, N. P. 1990, *The principles of design* Oxford University Press, New York.

230. Suh, N. P. 1998, "Axiomatic Design Theory for Systems", *Research in Engineering Design*, vol. 10, no. 4, pp. 189-209.

231. Sunnersjö, S. *Intelligent Computer systems for automated Engineering design*. 2006. Ph.D. Course, slide nr. 39 and 42 in lecture on "Introduction in automated design". 15-2-2006.
Ref Type: Slide

232. Takeuchi, H. & Quelch, J. A. 1983, "Quality Is More than Making a Good Product", *Harvard Business Review*, vol. 61, no. 4, pp. 139-145.

233. Taylor, F. W. 1911, *The Principles of Scientific Managment* Cosimo Classics, New York.

234. Teije, A. T., Harmelen, F. v., Schreiber, A. T., & Wielinga, B. J. 1998, "Construction of problem-solving methods as parametric design", *International Journal of Human Computer Studies*, vol. 49, no. 4, p. 363.

235. The Jensen Group 1997, *Changing how we work: The search for a simpler way*, The Jensen Group, Northern Illinois University College of Business.

236. Thompson, J. 1967, *Organizations in action* Transaction Publishers, New Brunswick, New Jersey.

237. Thoriacius, L. 2002, "A model of visual, aesthetic communication focusing on Web sites", *Digital Creativity*, vol. 13, no. 2, pp. 85-98.

238. Tichem, M. & Storm, T. 1997, "Designer support for product structuring--development of a DFX tool within the design coordination framework", *Computers in Industry*, vol. 33, no. 2-3, pp. 155-163.

239. Tjalve, E. 1979, *Systematic design of industrial products* Institute for Product Development, Technical University of Denmark, Kongens Lyngby, Denmark.

240. Tollenaere, M. & Jose, A. 2005, "Modular and platform methods for product family design: literature analysis", *Journal of Intelligent Manufacturing*, vol. 16, no. 3, pp. 371-390.

241. Ulrich, K. T. & Seering, W. P. 1990, "Function sharing in mechanical design", *Design Studies*, vol. 11, no. 4, pp. 223-234.

242. Ulrich, K. 1995, "The role of product architecture in the manufacturing firm", *Research Policy*, vol. 24, no. 3, pp. 419-440.

243. Ulrich, K. & Eppinger, S. D. 2004, *Product Design and Development, Third Edition* Irwin McGraw-Hill, Boston, Mass.

244. Ulrich, K. & Tung, K. 1991, "Fundamentals of product modularity", *Issues in Design/Manufacture Integration - 1991*, vol. 39, pp. 73-79.

245. Umeda, Y., Takeda, H., Tomiyama, T., & Yoshikawa, H. "Function, behaviour, and structure", J. S. Gero, ed., Comput. Mech. Publications, pp. 177-193.

246. Umeda, Y. & Tomiyama, T. 1997, "Functional reasoning in design", *IEEE Expert*, vol. 12, no. 2, pp. 42-48.

247. Uschold, M. 1996, "The use of the typed lambda calculus for guiding naive users in the representation and acquisition of part-whole knowledge", *Data and Knowledge Engineering*, vol. 20, no. 3, pp. 385-404.

248. Uschold, M. & Gruninger, M. 1996, "Ontologies: principles, methods and applications", *Knowledge Engineering Review*, vol. 11, no. 2, pp. 93-136.

249. van Hoek, R. I. 2001, "The rediscovery of postponement a literature review and directions for research", *Journal of Operations Management*, vol. 19, no. 2, pp. 161-184.

250. Van Wie, M., Bryant, C. R., Bohm, M. R., Mcadams, D. A., & Stone, R. B. 2005, "A model of function-based representations", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, vol. 19, no. 2, pp. 89-111.

251. Varzi, A. C. 1996, "Parts, wholes, and part-whole relations: The prospects of mereotopology", *Data and Knowledge Engineering*, vol. 20, no. 3, pp. 259-286.

252. VDI 1986, *VDI 2221 - Systematic approach to the design of Technical Systems and products*, VDI Society for Product Development, Design and Marketing, Dusseldorf, Germany, 2221.

253. Wacker, J. G. 1998, "A definition of theory: research guidelines for different theory-building research methods in operations management", *Journal of Operations Management*, vol. 16, no. 4, pp. 361-385.

254. Wacker, J. G. 2004, "A theory of formal conceptual definitions: developing theory-building measurement instruments", *Journal of Operations Management*, vol. 22, no. 6, pp. 629-650.

255. Wakefield, R. L. 2005, "Identifying knowledge agents in a KM strategy: the use of the structural influence index", *Information & Management*, vol. 42, no. 7, pp. 935-945.

256. Walsham, G. 2001, "Knowledge Management: The Benefits and Limitations of Computer Systems", *European Management Journal*, vol. 19, no. 6, pp. 599-608.

257. Watson, R. A. & Pollack, J. B. 2005, "Modular interdependency in complex dynamical systems", *Artificial Life*, vol. 11, no. 4, pp. 445-457.

258. Weaver, W. 1948, "Science and complexity," in *Facets of system science*, 2 edn, G. J. Klir, ed., Kluwer Academic / Plenum Publisher, New York, pp. 533-540.

259. Weber, M. 1924, "Legitimate authority and bureaucracy," in *Organiztion Theory: Selected readings*, 5 edn, Pugh & S. Derek, eds., Penguin Books, London.

260. Wexler, M. N. 2001, "The who, what and why of knowledge mapping", *Journal of Knowledge Management*, vol. 5, no. 3, pp. 249-264.

261. Wiener, N. 1948, *Cybernetics*, 2 edn, the MIT press, Cambridge, Massachusetts.

262. Wiig, K. M. 1997, "Knowledge Management: An Introduction and Perspective", *Journal of Knowledge Management*, vol. 1, no. 1, pp. 6-14.

263. Williams, B. C., Ingham, M. D., Chung, S., Elliott, P., Hofbaur, M., & Sullivan, G. T. 2003, "Model-Based Programming of Fault-Aware Systems", *AI Magazine*, vol. 24, no. 4, pp. 61-75.

264. Williams, B. C. & Pandurang Nayak, P. 1996, "Immobile robots: AI in the new millennium", *AI Magazine*, vol. 17, no. 3, pp. 16-35.

265. Wilson, T. L., Bostrom, U., & Lundin, R. 1999, "Communications and Expectations in After-Sales Service Provision : Experiences of an International Swedish Firm", *Industrial Marketing Management*, vol. 28, no. 4, pp. 381-394.

266. Winston, M. E., Chaffin, R., & Herrmann, D. 1987, "A taxonomy of part-whole relations", *Cognitive Science*, vol. 11, pp. 417-444.

267. Wise, R. & Baumgartner, P. 1999, "Go downstream: The new profit imperative in manufacturing", *Harvard Business Review*, vol. 77, no. 5, pp. 133-141.

268. Wooldridge, M. & Jennings, N. R. 1995, "Intelligent agents: theory and practice", *Knowledge Engineering Review*, vol. 10, no. 2, pp. 115-152.

269. Yoo, J., Catanio, J., Paul, R., & Bieber, M. 2004, "Relationship analysis in requirements engineering", *Requirements Engineering*, vol. 9, no. 4, pp. 238-247.

270. Zhang, W. Y., Tor, S. B., & Britton, G. A. "A functional modelling approach for modular product design", in *International Conference on Manufacturing Automation: Advanced Design and Manufacturing in Global Competition*, pp. 31-38.

271. Zhang, W., Mei, H., & Zhao, H. 2006, "Feature-driven requirement dependency analysis and high-level software design", *Requirements Engineering*, vol. 11, no. 3, pp. 205-220.